PACKT PUBLISHING

异步图书 www.epubit.com.cn

使用Python和Selenium进行Web自动化测试的实战指南

Selenium自动化测试
——基于Python语言

Learning Selenium Testing Tools with Python

[印度] 昂米沙·冈迪察（Unmesh Gundecha） 著
金鑫 熊志男 译
testwo测试窝 审

中国工信出版集团 人民邮电出版社 POSTS & TELECOM PRESS

□ □

□□□□

□□□□

□□□□

□□□

□□□

□□□□

□□□

□□□□□

□□

□1□ □□Python□Selenium WebDriver□□
1.1 □□□□
1.1.1 □□Python

1.1.2 □□Selenium□
1.1.3 □□Selenium WebDriver Python□□
1.1.4 □□□□IDE
1.1.5 PyCharm□□
1.2 □□□Selenium Python□□
1.3 □□□□□□
1.3.1 □□IE□□□
1.3.2 □□Google Chrome□□□
1.4 □□□□

□2□ □□unittest□□□□□□
2.1 unittest□□□□□□
2.1.1 TestCase□
2.1.2 □□□□setUp()□□□tearDown()□□
2.1.3 □□
2.1.4 □□□□
2.2 □□HTML□□□□□□□
2.3 □□□□

□3□ □□□□
3.1 □□□□□□□□□□□□□□
3.1.1 □□□□□□□Firebug□□□□□□□□
3.1.2 □□□Chrome□□□□□□□□□
3.1.3 □IE□□□□□□□□□
3.2 □□□□
3.2.1 ID□□

3.2.2 name□□
3.2.3 class□□
3.2.4 tag□□
3.2.5 XPath□□
3.2.6 CSS□□□□□
3.2.7 Link□□
3.2.8 Partial link□□
3.3 □□□□
3.4 □□□□

□4□ Selenium Python API□□
4.1 HTML□□□□
4.2 WebDriver□□
4.2.1 WebDriver□□
4.2.2 WebDriver□□
4.3 WebElement□□
4.3.1 WebElement□□
4.3.2 WebElement□□
4.4 □□□□□□□□□□□□□□□□□□
4.4.1 □□□□□□□□□□□
4.4.2 □□□□□□□□
4.4.3 is_selected()□□
4.4.4 clear()□send_keys()□□
4.5 □□□□□□
4.5.1 Select□□
4.5.2 Select□□

4.5.3 Select□□
4.6 □□□□□□□□
4.6.1 Alert □□
4.6.2 Alert□□
4.6.3 Alert□□
4.6.4 □□□□□□□□
4.7 □□□□

□5□ □□□□□□
5.1 □□□□
5.2 □□□□
5.3 expected_conditions□
5.3.1 □□□□□□□□□□
5.3.2 □□□□□□Alerts
5.4 □□□□□□□□□
5.5 □□□□

□6□ □□□□□□
6.1 Selenium Standalone Server
6.1.1 □□Selenium Standalone Server
6.1.2 □□Selenium Standalone Server
6.2 □Selenium Standalone Server□□□□
6.2.1 □□IE□□
6.2.2 □□Chrome□□
6.3 Selenium Grid
6.3.1 □□hub

6.3.2 □□□□
6.4 Mac OS X□Safari□□
6.5 □Grid□□□□□
6.6 □□□□□□□
6.7 □□□□

□7□ □□□□□
7.1 □□Appium
7.1.1 Appium□□□□□□□
7.1.2 Appium□□□□
7.2 □□Appium
7.3 iOS□□
7.4 Android□□
7.5 □□Sauce Labs
7.6 □□□□

□8□ Page Object□□□□□□□
8.1 □□□□□□
8.2 □□ddt□□□□□□□□
8.2.1 □□ddt
8.2.2 □□□□□□□□□□□□□□
8.3 □□□□□□□□□□□□□□
8.3.1 □□CSV□□□□
8.3.2 □□Excel□□□□
8.4 Page Object□□□□
8.4.1 □□□□

8.4.2 BasePage□□
8.4.3 □□Page Object
8.4.4 □□Page Object□□□□□□
8.5 □□□□

□9□ Selenium WebDriver□□□□□
9.1 □□□□□□□
9.1.1 □□□□
9.1.2 □□□□
9.2 □□JavaScript
9.3 □□□□
9.4 □□□□
9.5 □□□□□□
9.6 □□cookies
9.7 □□□□

□10□ □□□□□□□□□□□
10.1 □□□□□□□BDD□
10.1.1 Behave□□
10.1.2 □□□feature
10.2 □□□□Jenkins
10.2.1 Jenkins□□□□
10.2.2 □□Jenkins
10.3 □□□□

□□□□□□□□□

[illegible]

[illegible]Selenium[illegible]——[illegible] Python [illegible]

ISBN[illegible]978-7-115-46174-2

[illegible]

[illegible]

[illegible]

[illegible]

- [illegible] Unmesh Gundecha

□ □□□

□□□□ □ □

- □□□□□□□□□□□□ □□□□□□□□□□□11□

□□ 100164 □□□□

315@ptpress.com.cn

□□ http://www.ptpress.com.cn

- □□□□□□□(010)81055410

□□□□□□(010)81055315

[illegible]

Copyright © Packt Publishing 2014. First published in the English language under the title“Learning Selenium Testing Tools with Python”[illegible] ISBN 978-1-78398-350-6.

All rights reserved.

[illegible]Packt Publishing[illegible]

[illegible]

[illegible]

Selenium[illegible]Web[illegible]Python[illegible]Selenium WebDriver[illegible]Python [illegible] Selenium WebDriver [illegible]Selenium Python[illegible]unittest [illegible]HTML[illegible]Selenium Python API[illegible]iOS[illegible]Android[illegible]Page Object[illegible]Selenium WebDriver[illegible]

[illegible]

[illegible]

[illegible]BQConf[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]Java[illegible]C#[illegible]

[illegible]

[illegible]1[illegible]Selenium WebDriver[illegible]Web[illegible]Selenium WebDriver[illegible]Web[illegible]

[illegible]2[illegible]Python[illegible]Python[illegible]API[illegible]

[illegible]BDD[illegible]

[illegible]

[illegible]ThoughtWorks[illegible]QA Lead

[illegible]

[illegible]Web[illegible]Selenium[illegible]Selenium[illegible]Web[illegible]

[illegible]WebDriver[illegible]Server[illegible]Python[illegible]Selenium WebDriver[illegible]*Learning Selenium Testing Tools with Python*[illegible]Web[illegible]

[illegible]“[illegible]”[illegible]“[illegible]”[illegible]

[illegible]Unmesh Gundecha[illegible]2012[illegible]*Selenium Testing Tools Cookbook*[illegible]2015[illegible]

[illegible]“[illegible]”[illegible]

[illegible]

PK[illegible]

[illegible]

zhangtao@ptpress.com.cn[illegible]

[illegible]

[illegible]

[illegible]Web [illegible]

Python [illegible]

Selenium [illegible] UI [illegible]

[illegible] Selenium [illegible] UI [illegible]

——Ping++ [illegible]

Unmesh Gundecha[illegible]*Selenium Testing Tools Cookbook*[illegible]“Selenium[illegible]”[illegible]

□□□□□□□□□WebDriver□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□

□□□□“□□”□Python□□□□□□□“□□”□□□□□□□□□□□□□□□□□□□□□□WebDriver□□□□□□□□

——□□□ □□□□□□□□□□10□□□□□□□□□□□□□□□□□□□□□□□□□ITSM□PLM□□□□□□□□□□□□□□□□□□IT□□

[illegible]

Unmesh Gundecha[illegible]12[illegible]

[illegible]Ruby[illegible]Java[illegible]iOS[illegible]Android[illegible]PHP[illegible]

[illegible]

[illegible]

[illegible]Yuri Weinstein[illegible]

□□□□□

Adil Imroz□□□Python□□□

Dr. Philip Polstra□□□□□□□□□Dr. Phil□□□□□□□□□□□□□□□□□□□□□□□□□DEFCON□Black Hat□44CON□Maker Faire□□□□□□□□□□□□□□□□□□□□□□□□

Dr. Polstra□□□

Walt Stoneburner□□□□□□□□□□□□□□□□□□□□25□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

[illegible]
ASCII[illegible]
[illegible]
[illegible]“[illegible]”[illegible]

Yuri Weinstein[illegible]20[illegible]
[illegible]
[illegible]Ceph[illegible]

□□

Selenium□□□□□□□Web□□□Firefox□IE□Chrome□Safari□□Opera□□

Selenium□□□□□□□□□□□

- Selenium IDE□□□□□Firefox□□□□□□□□□□Firefox□□□□□□Selenium □□Firefox□□□□□□□□□□□□□□□□□□□□Selenium WebDriver□□□□□□□□□□□□□□□□□□□□□□□
- Selenium WebDriver□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□API□□□□□□□□□□□□□□□

□□□□□Linux□Windows□□Mac OS X□□□□□
□□□□□□□□□□□□□□□WebDriver□□□Java□C#□
Python□Ruby□PHP□JavaScript□□□□□□□□
□□□□□□□□□Web□□□□□□□□□□□

- Selenium Standalone Server□□□□□□□□□
□□Selenium Grid□□□□□□□□□□□□□□□□□□□
Selenium □□□□□□□□□□□□□□□□Grid□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□Android□iOS□

□□□□□□□□□□□□□□□□□□Python□□□□
Selenium WebDriver□□□□□□□□Web□□□□□□□
□□□□□□□□□□□□Selenium□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
Python□□□□□

□□□□

（1）安装Python和Selenium WebDriver。安装Python和Selenium WebDriver是使用Python进行自动化测试的前提条件。

（2）安装unittest测试框架。unittest是Python的一个单元测试框架，可以用来编写和运行测试用例，并生成测试报告。

（3）编写测试用例。根据测试需求，编写测试用例，使用Selenium提供的API来模拟用户操作，如点击、输入、选择等，并使用XPath或CSS选择器来定位元素。

（4）Selenium Python API。Selenium Python API提供了WebDriver的各种方法，如JavaScript 执行、处理frames等，可以根据测试需求选择合适的方法。

[illegible]5[illegible] [illegible]

Selenium[illegible]

[illegible]

[illegible]6[illegible] [illegible]

Selenium Grid[illegible]Remote WebDriver[illegible]

[illegible]Selenium Grid[illegible]

[illegible]PhantomJS[illegible]UI[illegible]

[illegible]Sauce Labs [illegible]

BrowserStack[illegible]

[illegible]7[illegible] [illegible]Selenium WebDriver[illegible]

Appium[illegible]iOS[illegible]Android[illegible]Android[illegible]

[illegible]App[illegible]

[illegible]8[illegible]Page Object[illegible] [illegible]

[illegible]Page Object

[illegible]

[illegible]UI[illegible]unittest[illegible]

[illegible]9[illegible]Selenium WebDriver[illegible] [illegible]cookies[illegible]

[illegible]10[illegible] [illegible]Selenium[illegible]“[illegible]Selenium[illegible]UI[illegible]Jenkins[illegible]”[illegible]

[illegible]Python[illegible]Selenium WebDriver[illegible]Web[illegible]

[illegible]

[illegible]Python[illegible]Web[illegible]HTML[illegible]JavaScript[illegible]CSS[illegible]XML[illegible]Python[illegible]1[illegible]Python[illegible]

[illegible]

[illegible]QA[illegible]Web[illegible]Python[illegible]Selenium WebDriver[illegible]Web[illegible]Selenium[illegible]Python[illegible]Selenium WebDriver[illegible]

[illegible]

[illegible]

[illegible]

```
# create a new Firefox session
driver = webdriver.Firefox()
driver.implicitly_wait(30)
driver.maximize_window()
```

[illegible]

```
# run the suite
xmlrunner.XMLTestRunner(verbosity=2,output='test-reports').
 run(smoke_tests)
```

[illegible]\[illegible]

```
pip install -U selenium
```

[illegible]“[illegible]**Tools**[illegible]

Internet Options”[illegible]

> [illegible]

> [illegible]

[illegible]

[illegible]——[illegible]

[illegible]contact@epubit.com.cn[illegible]
[illegible]

[illegible]
[illegible]zhangtao@ptpress.com.cn[illegible]

第1章　使用Python和Selenium WebDriver

Selenium是一个用于Web应用程序测试的工具。Selenium测试直接运行在浏览器中，就像真正的用户在操作一样。支持的浏览器包括IE（7, 8, 9, 10, 11），Mozilla Firefox，Safari，Google Chrome，Opera等。这个工具的主要功能包括：测试与浏览器的兼容性——测试你的应用程序能否很好得工作在不同浏览器和操作系统之上。测试系统功能——创建回归测试检验软件功能和用户需求。支持自动录制动作和自动生成 Selenium 脚本。Selenium WebDriver是一种新的自动化测试工具，它对不同浏览器提供了一个统一的接口，用来驱动浏览器。它的底层实现是直接调用浏览器厂商提供的组件接口，这样可以让浏览器感觉自己是被用户操作，从而执行更为纯粹的测试。这种实现方式和Selenium client library那种在页面中注入脚本的方式有很大不同。

本章将介绍如何使用Selenium client library和Python来编写测试脚本。Python是一种面向对象的解释型计算机程序设计语言，具有丰富和强大的库。它常被昵称为胶水语言，能够把用其他语言制作的各种模块（尤其是C/C++）很轻松地联结在一起。常见的一种应用情形是，程序员用Python快速生成程序的原型（有时甚至是程序的最终界面），然后对其中有特别要求的部分，用更合适的语言改写，比如3D游戏中的图形渲染模块，性能要求特别高，就可以用C/C++重写，而后封装为Python可以调用的扩展类库。需要注意的是在您使用扩展类库时可能需要考虑平台问题，某些可能不提供跨平台的实现。

□□Python□Selenium WebDriver client library□□□□Selenium WebDriver□□□□□□□□Selenium standalone server□□□□□□□□□□B/S□□□Selenium language bindings□□□□□□David Burns□Adam Goucher□Maik Röder□Jason Huggins□Luke Semerau□Miki Tebeka□Eric Allenin□

Selenium WebDriver client library □□□□Python□□□2.6□2.7□3.2□3.3□

□□□□□□□Python□Selenium WebDriver client library□□□□□□□□□□□□□□□

□□□□□□□□□

- □□Python□Selenium□□
- □□□□□Python□□□□
- Selenium WebDriver□□Python□□□□□□

- □□□□IE□Chrome□□□□□□□□

1.1 □□□□

□□□□□□□□Python□Selenium□□

1.1.1 □□Python

□□□□Linux□□□Mac OS X□□□□□UNIX□□□□□□□□Python□□□□□□□□□□□□□Windows□□□□□□□□□□□□Python□□□□□□□□□Python□□□□□□□□□□□□□□□□□http://python.org/download/□

> □□□□□□□□□□□Python 2.7□Python 3.0□□□□□Windows 8 □□□□□□□□□

1.1.2 □□Selenium□

Selenium□□□□□□□Selenium WebDriver Python client library□□□□□□Selenium □□□□□□□□pip□□□□□https://pip.pypa.io/en/latest/□

□□pip□□□□□□□□□□□□□□□□□□□□□Selenium□□□□

```
pip install -U selenium
```

□□□□□□□□□□□□□□□□Selenium WebDriver client library□□□□□□□□□□□□Python□□□□□□□□□□□□□□□□□□pip□□□□□□□□□□□Selenium□□□□□□□□□□□□□□□□□ -U □□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□Selenium□□□□https://pypi.python.org/pypi/selenium□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

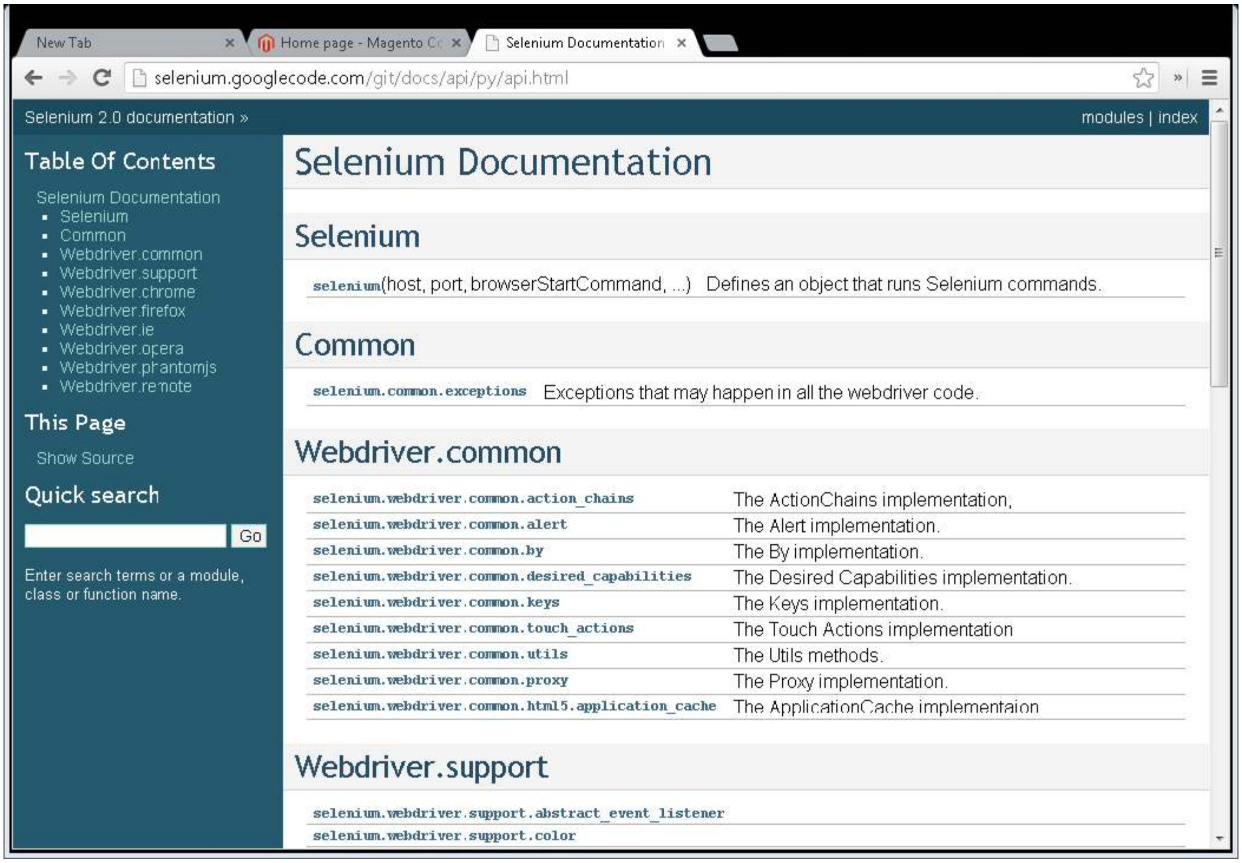
```
python setup.py install
```

1.1.3 □□Selenium WebDriver Python□□

Selenium WebDriver Python client library □□□□□□□□□□□□

http://selenium.googlecode.com/git/docs/api/py/api.html

□□□□□□



□□□□□Selenium WebDriver□□□□□□□□□□□□□□□□□□□□□□□Selenium□□□□□□□□□

- □□□□□http://docs.seleniumhq.org/docs/□□□□□□Selenium□□□□□□□□□□□□□□□□□□□□□□□□□□□
- Selenium Wiki□□□https://code.google.com/p/selenium/w/li

st[illegible]

1.1.4 [illegible]IDE

[illegible]Python[illegible]Selenium WebDriver[illegible]IDE[illegible]**IDE**[illegible]Python[illegible]Emacs[illegible]Vim[illegible]Notepad[illegible]IDE[illegible]IDE[illegible]

- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]

□□□□□Python□□□□□□□□□□□□□Python□□□□□□□□□□□□□□□□□□□□□□□□□□□IDE□

□□□□□□□□□□□□Python□□□□□□□□IDE□□□□□□□□□□□□□□□□□

1.1.4.1 PyCharm

PyCharm□JetBrains□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□IntelliJ IDEA□RubyMine□PhpStorm□TeamCity□

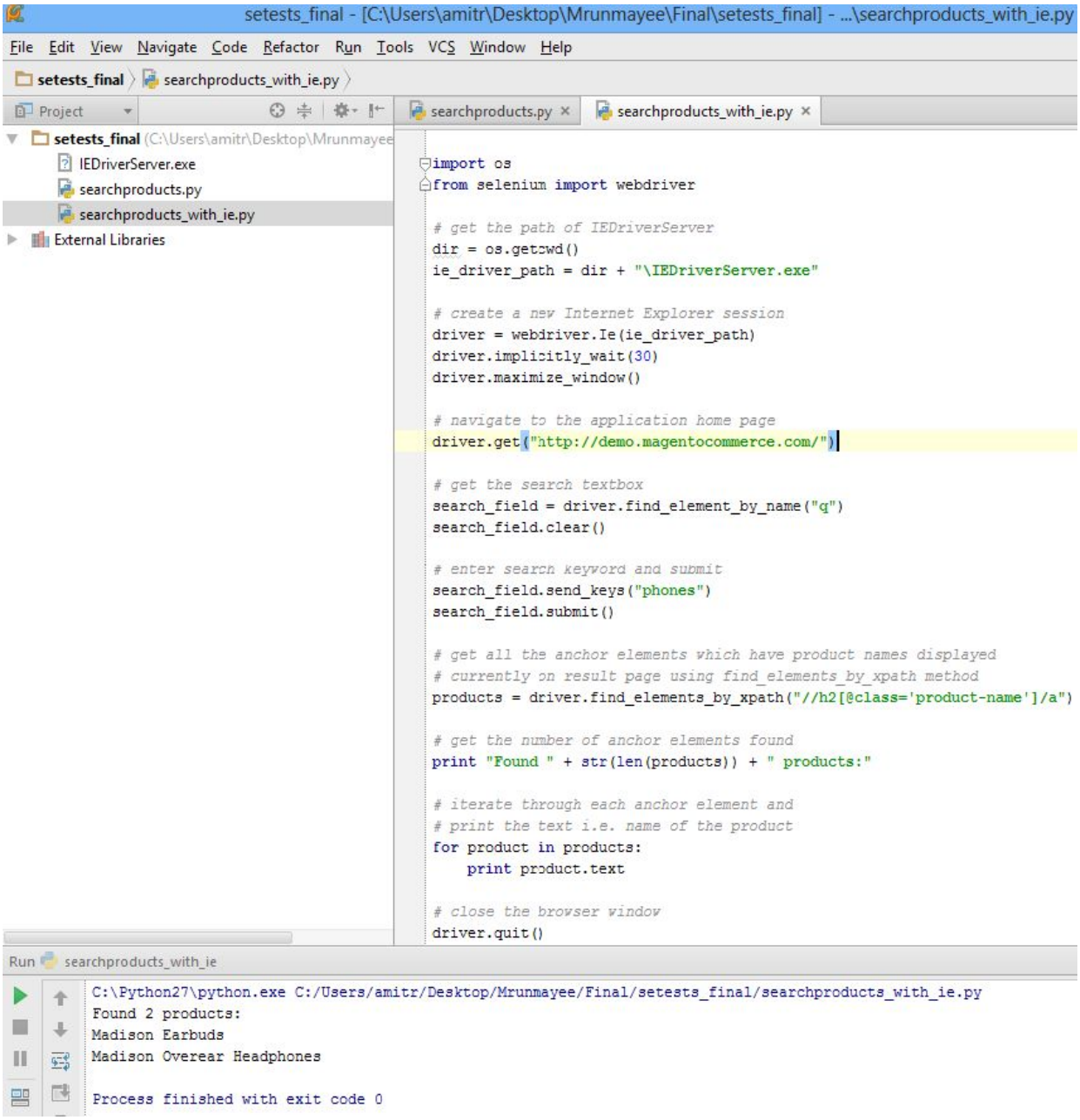
PyCharm□□□□□□□□□□□□□□□□□□□□□□□□IDE□□□□□JetBrains □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

PyCharm□□Windows□□□Linux□□□Mac□□□□□□□□□PyCharm□□□□□□□□□□□□□http://www.jetbrains.com/pycharm/□

PyCharm[illegible]——[illegible]PyCharm[illegible]Selenium[illegible]

[illegible]Selenium[illegible]PyCharm[illegible]PyCharm[illegible]Selenium[illegible]

> [illegible]PyCharm[illegible]IDE[illegible]



1.1.4.2 PyDev Eclipse plugin

PyDev Eclipse plugin□□□□□Python□□□□

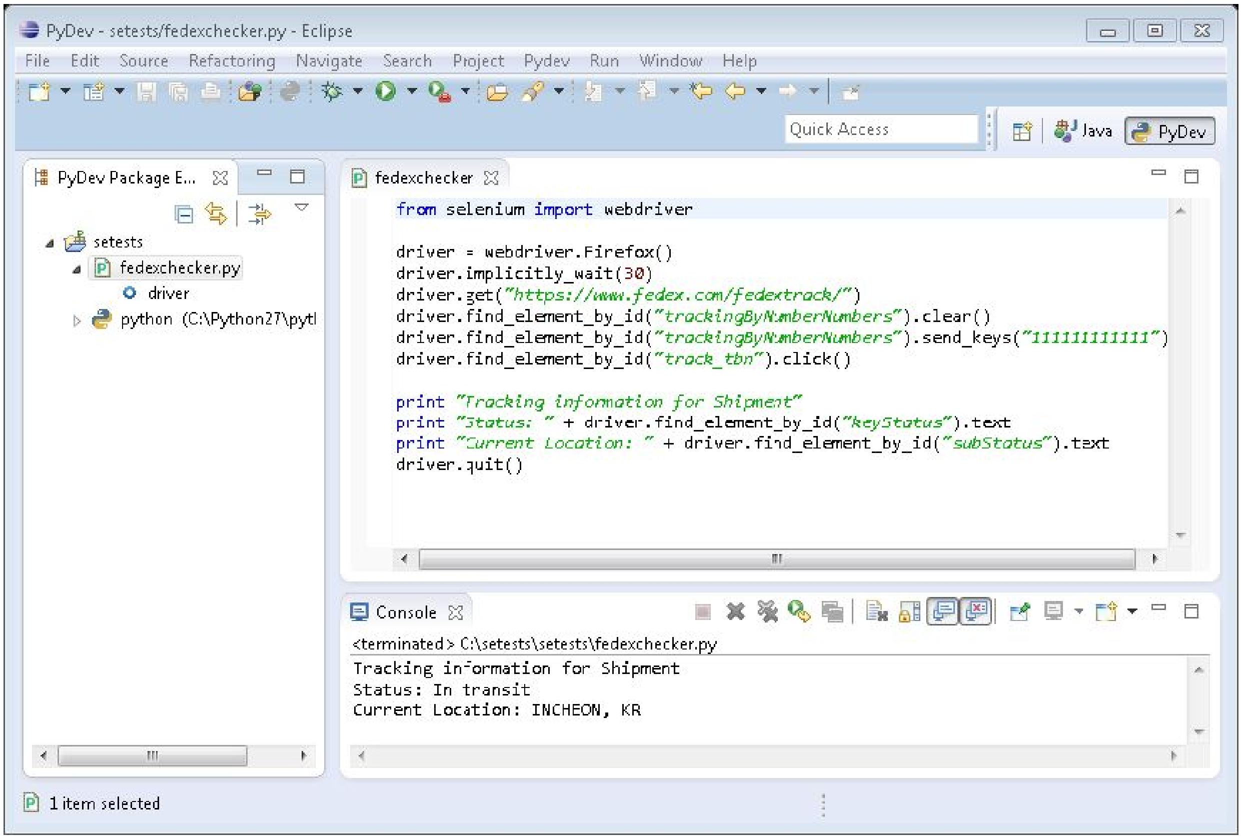
□□□□□□□□□□□□Eclipse□□□□□□□□□□□□□□□□□□□□□

□□Java□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Eclipse□□□□□□□IDE□□□Windows□□□ Linux□□□Mac□□□□□□□□□□□□□Eclipse□□□□□□ http://www.eclipse.org/downloads□

PyDev plugin□□□□□□Eclipse□□□□□□□□□□ Lars Vogel□□□□□□□□□□PyDev□ http://www.vogella.com/tutorials/Python/article.html□□□□□□□□□□□□□□□□□□ http://pydev.org/□

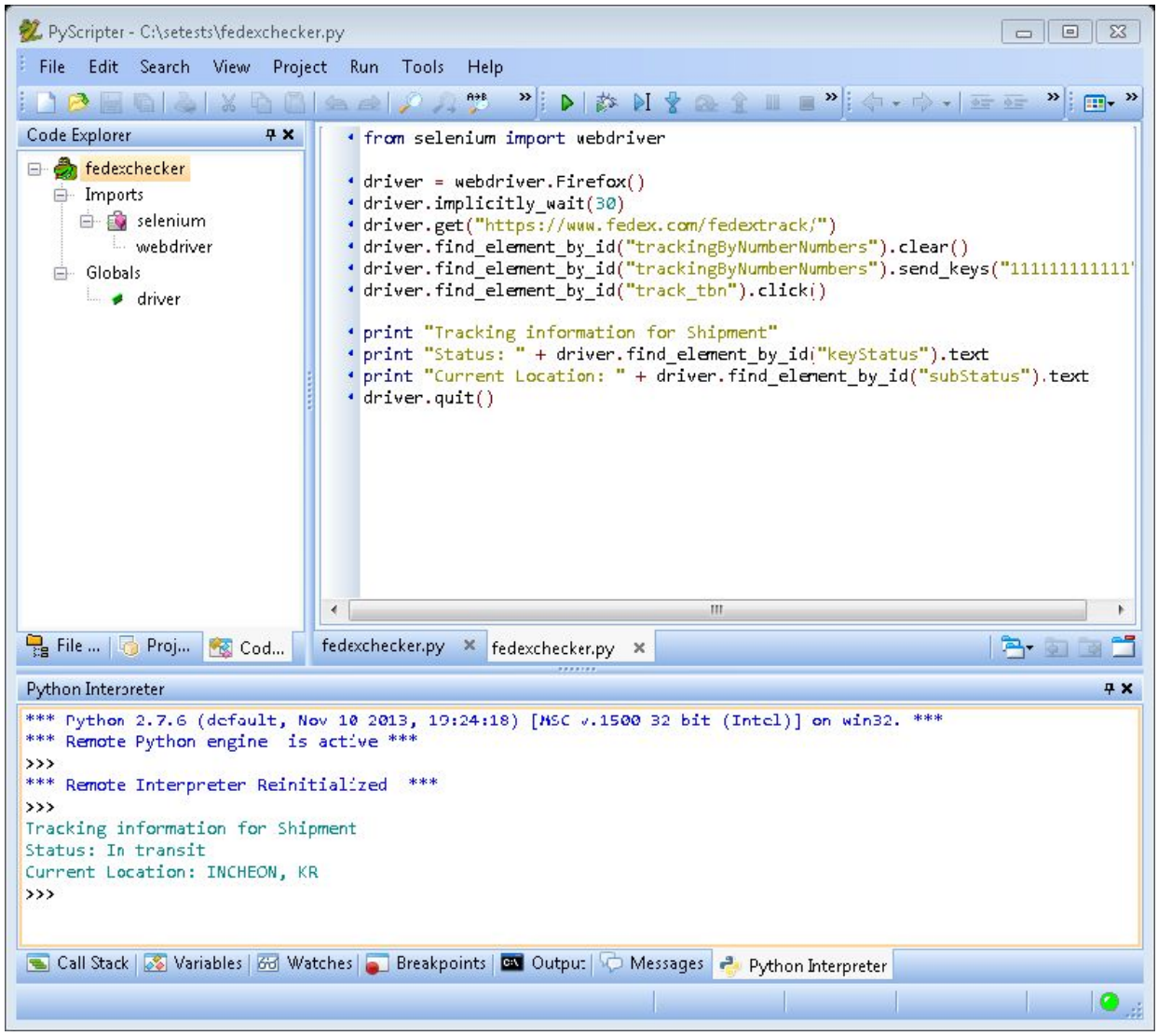
□□□□□PyDev Eclipse plugin □□□□□□ Selenium□□□□□□□□



1.1.4.3 PyScripter

□□Windows□□□□□PyScripter□□□□□□□□□□□□□□□□□□□□□□□IDE□PyScripter□□□□□IDE□□https://code.google.com/ p/pyscripter/□

□□□□ PyScripter□□□□□□□ Selenium □□□□ □□□□



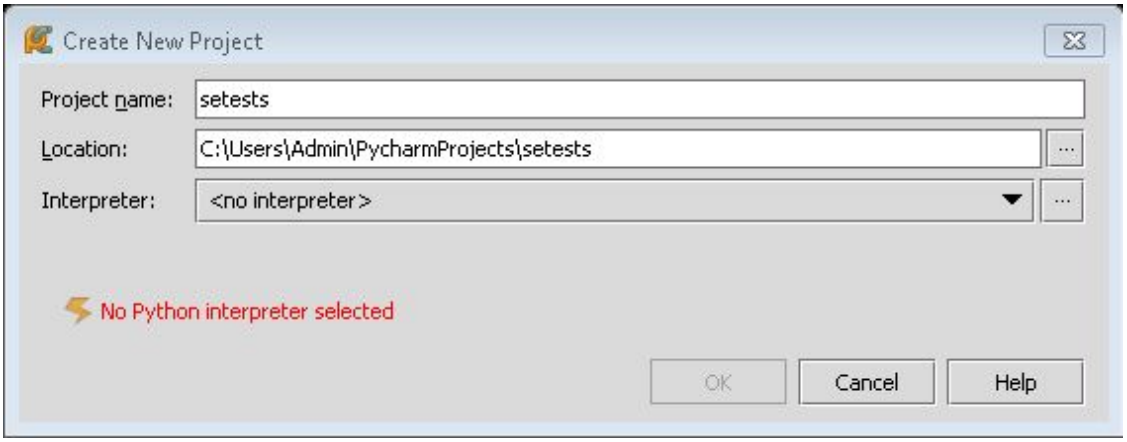
1.1.5 PyCharm□□

□□□□□□□□□□□□IDE□□□□□PyCharm□□□□□□□□□□□□□□□□□□□□□□PyCharm□□□□□□□□□□□□□□□□□□IDE□□□□□□□□□□□□□□□□□□□□□□□□□□□□PyCharm□□□□□□Selenium Python□□□□

□1□□JetBrains□□□□□□□□PyCharm□

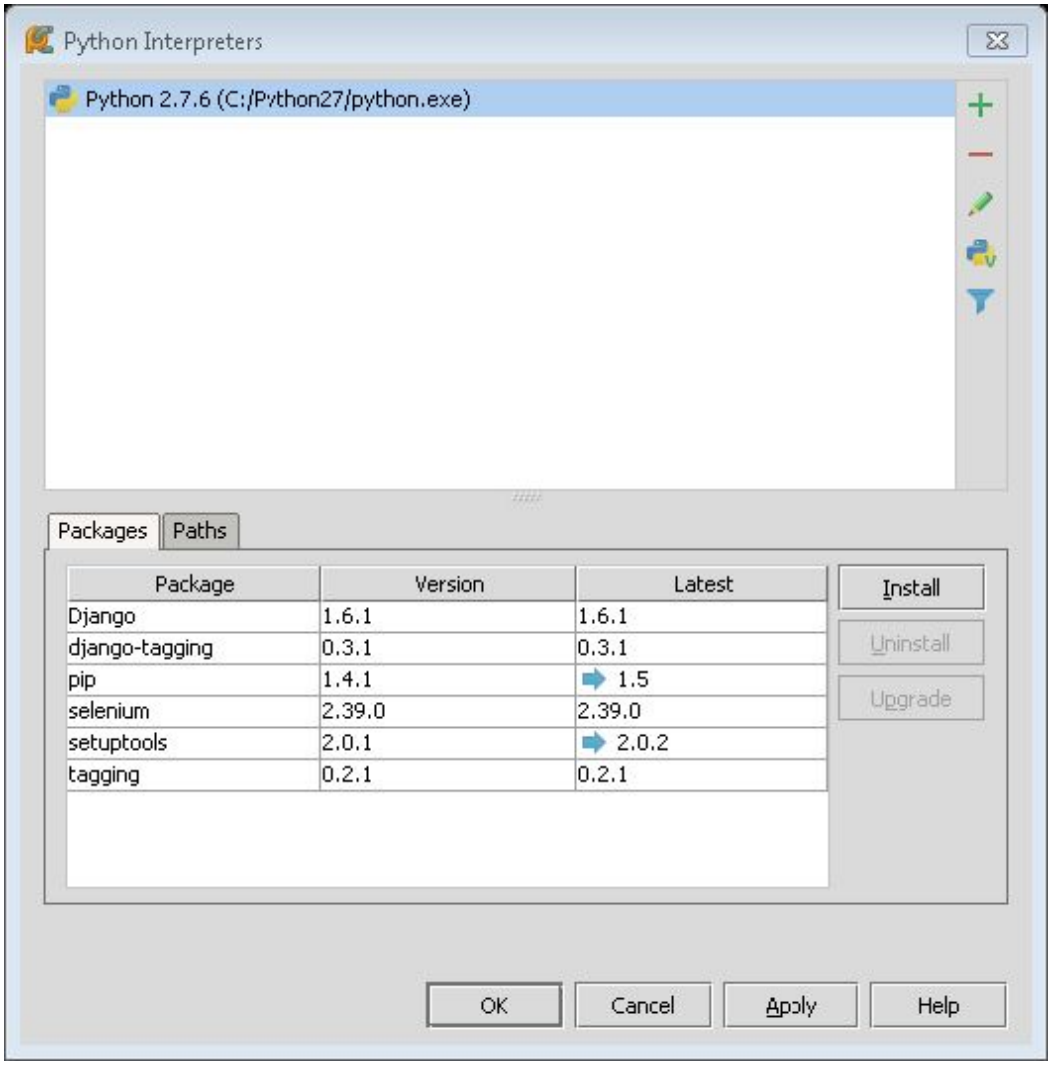
□2□□□PyCharm□□□□□□□□□□□□□□**Create New Project**□□□

（3）在Create New Project窗口，设置新建项目的
Project name为setests，
设置项目在PyCharm中的保存路径，
Interpreter为[...]默认设置，

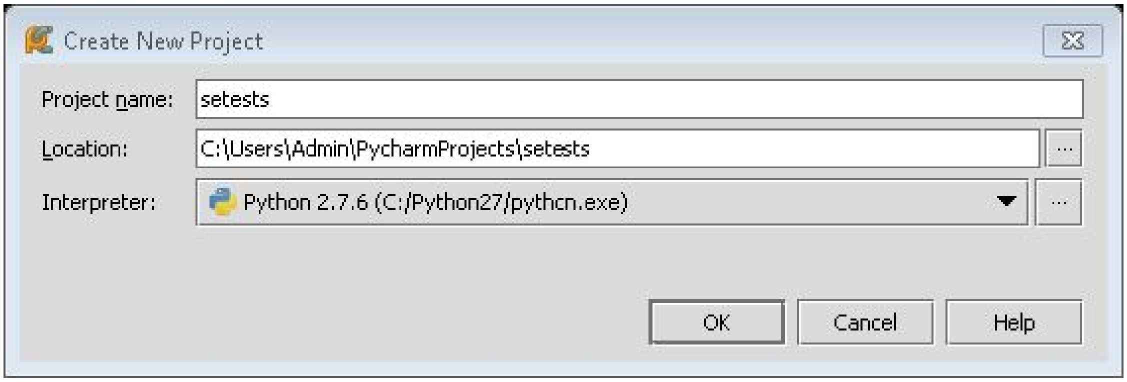


（4）点击右侧**Python Interpret**配置PyCharm项目需要的Python解释器，Select
Interpreter Path默认设置，

（5）PyCharm 进入设置界面，选择 Project Interpreter，点击 Packages，选择 Python 版本，点击 **Apply** 按钮，点击 **OK** 按钮



（6）点击 **Create New Project** 按钮，点击 **OK** 按钮



1.2 [illegible]Selenium Python[illegible]

[illegible]Selenium WebDriver[illegible]Python[illegible]Selenium WebDriver[illegible]

[illegible]Web[illegible]Web[illegible]**Magento**[illegible]:http://demo.magentocommerce.com/[illegible]

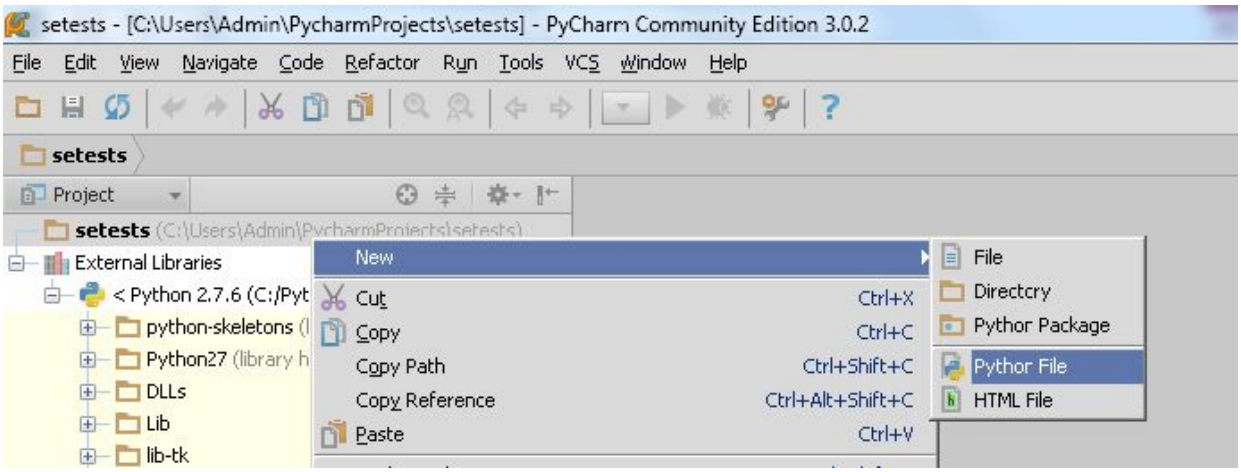
> [illegible]

□□□□□http://packtpub.com□□□http://www.packtpub.com/support□□□□□□□□□□□□□□□□□□□□□□□□□

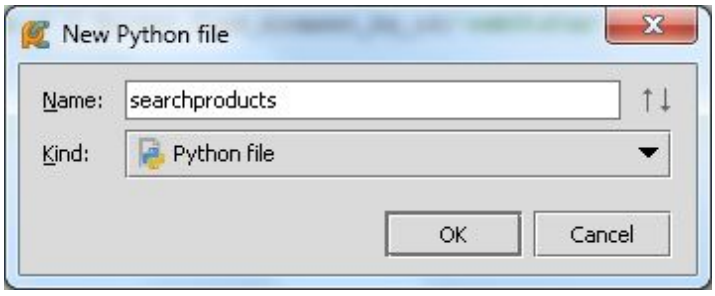
□□□□□□□□□github□□□□□□□□https://github.com/upgundecha/learnsewithpython□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□1□□□□□□□□PyCharm□□□□□□□□□□□Selenium WebDriver client library□Python□□□□□□□□□□□□`setests`□□□□□□□□**New→Python File**□



□2□□**New Python file**□□□□□□□□□□□□□□□□□“searchproducts”□□□□□OK□□□



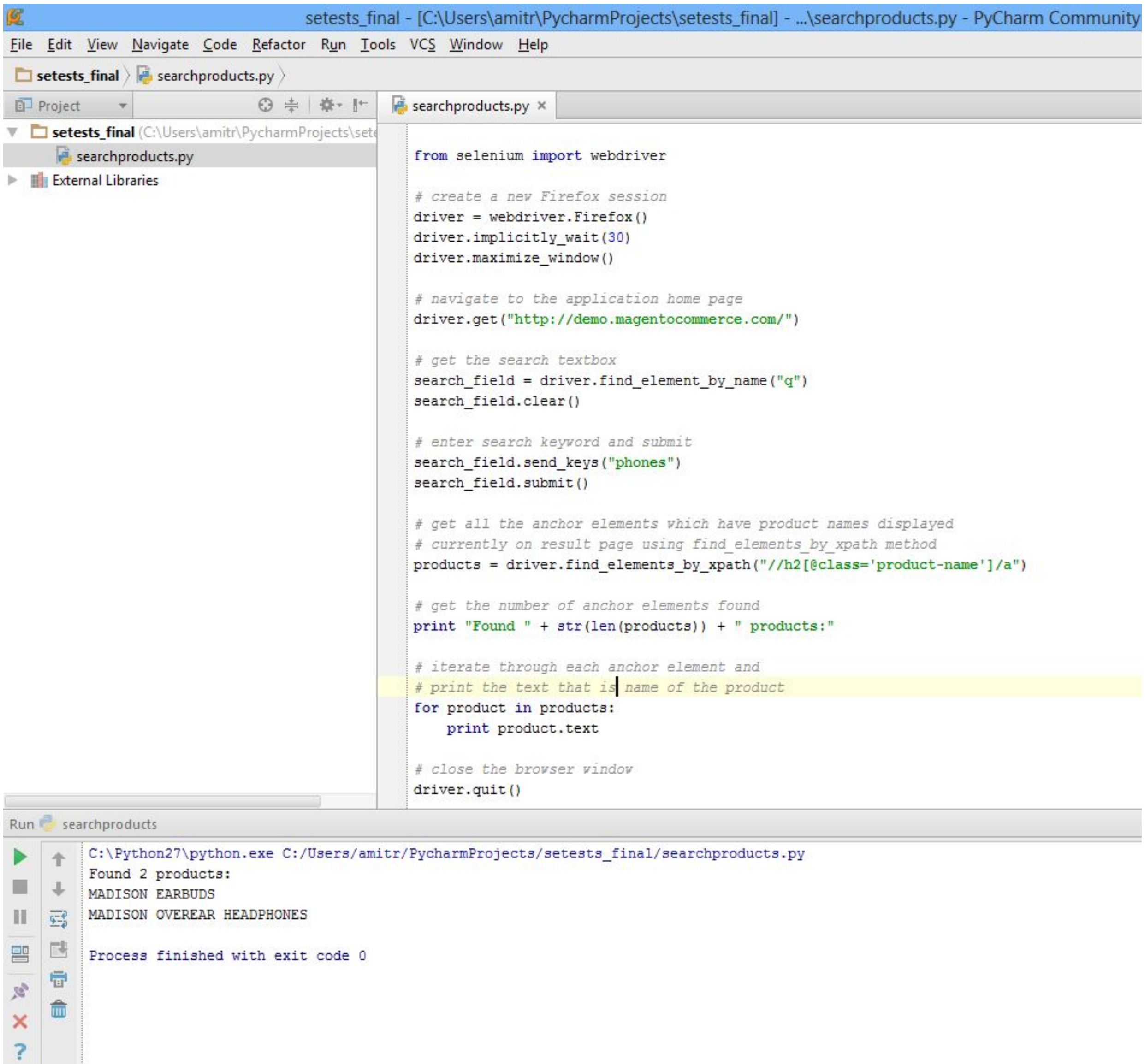
□3□PyCharm□□□□□□□□□□□□□□□searchproducts.py□□□□□□□□□□□□□searchproducts.py□□

```
from selenium import webdriver

# create a new Firefox session
driver = webdriver.Firefox()
```

```
driver.implicitly_wait(30)
driver.maximize_window()

# navigate to the application home page
driver.get("http://demo.magentocommerce.com/")

# get the search textbox
search_field = driver.find_element_by_name("q")
search_field.clear()

# enter search keyword and submit
search_field.send_keys("phones")
search_field.submit()

# get all the anchor elements which have product
names displayed
# currently on result page using
find_elements_by_xpath method
products =
driver.find_elements_by_xpath("//h2[@class='produc
t-name']/a")

# get the number of anchor elements found
print "Found " + str(len(products)) + " products:"

# iterate through each anchor element and print
the text that is
# name of the product
for product in products:
   print product.text

# close the browser window
driver.quit()
```

> □□□□□□□□□□IDE□□□□□□□□□□□□□□□□□□□
> □□□□□□□□□□□□□searchproducts.py□

□4□□□□□□□□□□□□□□□□PyCharm□□□□□□□□□□Ctrl + Shift+ F10□□□**Run**□□□□□**Run 'searchproducts'**□□□□□□□□□□□□□□□□□□□Firefox□□□□□□□□□□□□□□□Firefox□□□□□□□□□□□□□Selenium□□□□□□□□□□□□□□□□□□□Firefox□□□□□□□□□□□□□□□□□□PyCharm□□□□□□□□□□□□□


setests_final - [C:\Users\amitr\PycharmProjects\setests_final] - ...\searchproducts.py - PyCharm Community
File Edit View Navigate Code Refactor Run Tools VCS Window Help
setests_final searchproducts.py
Project
setests_final (C:\Users\amitr\PycharmProjects\sete
searchproducts.py
External Libraries
searchproducts.py
from selenium import webdriver
# create a new Firefox session
driver = webdriver.Firefox()
driver.implicitly_wait(30)
driver.maximize_window()
# navigate to the application home page
driver.get("http://demo.magentocommerce.com/")
# get the search textbox
search_field = driver.find_element_by_name("q")
search_field.clear()
# enter search keyword and submit
search_field.send_keys("phones")
search_field.submit()
# get all the anchor elements which have product names displayed
# currently on result page using find_elements_by_xpath method
products = driver.find_elements_by_xpath("//h2[@class='product-name']/a")
# get the number of anchor elements found
print "Found " + str(len(products)) + " products:"
# iterate through each anchor element and
# print the text that is name of the product
for product in products:
print product.text
# close the browser window
driver.quit()
Run searchproducts
C:\Python27\python.exe C:/Users/amitr/PycharmProjects/setests_final/searchproducts.py
Found 2 products:
MADISON EARBUDS
MADISON OVEREAR HEADPHONES
Process finished with exit code 0

□□□□□□□□□□□□□□□□□□□□□□□□□□

setests□□□□□□□□□□□□□□□□

```
python searchproducts.py
```

[illegible]

[illegible]Selenium WebDriver[illegible]

Selenium.webdriver[illegible]Selenium[illegible]Firefox[illegible]Chrome[illegible]IE[illegible]Safari[illegible]RemoteWebDriver[illegible]

[illegible]Selenium[illegible]WebDriver[illegible]Selenium WebDriver[illegible]

```
from selenium import webdriver
```

[illegible] Selenium [illegible] Firefox [illegible] Firefox [illegible]

```
driver = webdriver.Firefox()
```

[illegible] Firefox [illegible]

```
driver.implicityly_wait(30)
driver.maximize_window()
```

[illegible] 30 [illegible] Selenium [illegible] Selenium API [illegible] 5 [illegible]“[illegible]”[illegible]

[illegible] URL [illegible] driver.get() [illegible] get() [illegible] WebDriver [illegible]

[illegible]Selenium[illegible]

[illegible]

[illegible]**Search**[illegible]HTML[illegible]

Selenium[illegible]Selenium

WebDriver[illegible]

[illegible]3[illegible]“[illegible]”[illegible]

[illegible]find_element_by_name[illegible]

[illegible]name[illegible]

[illegible]HTML[illegible]

[illegible]

[illegible]1[illegible]q[illegible]name[illegible]

[illegible]

```
search_field = driver.find_element_by_name("q")
```

[illegible]2[illegible]clear()[illegible]

[illegible]send_keys()[illegible]

[illegible]submit()[illegible]

```
search_field.clear()
search_field.send_keys("phones")
search_field.submit()
```

□3□□□□□□□□□Firefox□□□□□□□□□□□□□□□□□□□□□□□phones□□□□□□□□□□□□□□□□□□□□□□find_elements_by_xpath□□□□□□□<a>□□□□□□□□□□□□□□□□□1□□□□□□

```
products =
  driver.find_elements_by_xpath("//h2[@class=
  'product-name']/a")
```

□4□□□□□□□□□□□□□□□□□□□□□□□□<a>□□□□□□□□□□□□□□□□<a>□□□text□□□□□

```
print "Found " + str(len(products)) + " products:"

for product in products:
    print product.text
```

□5□□□□□□□□□□□driver.quit()□□□□□Firefox□□□□

```
driver.quit()
```

[illegible]Selenium WebDriver[illegible]Python[illegible]Python[illegible]

1.3 [illegible]

[illegible]Firefox[illegible]Selenium[illegible]IE[illegible]Google Chrome[illegible]Safari[illegible]Opera[illegible]PhantomJS[illegible]UI[illegible]IE[illegible]Google Chrome[illegible]Selenium WebDriver[illegible]

1.3.1 [illegible]IE[illegible]

[illegible]IE[illegible]InternetExplorerDriver[illegible]

InternetExplorerDriver□□□□□□□□□□□□□□□□
WebDriver□□□□□□WebDriver□□□□□□□□IE□□□
□□□InternetExplorerDriver□□Windows XP□
Vista□Windows 7□Windows 8□□□□□□□□IE□
□□□□□□□□□□InternetExplorerDriver□

□1□□
http://www.seleniumhq.org/download/□□□
InternetExplorerDriver□□□□□□□□□□□□□□□□□
□□□32□□64□□□□

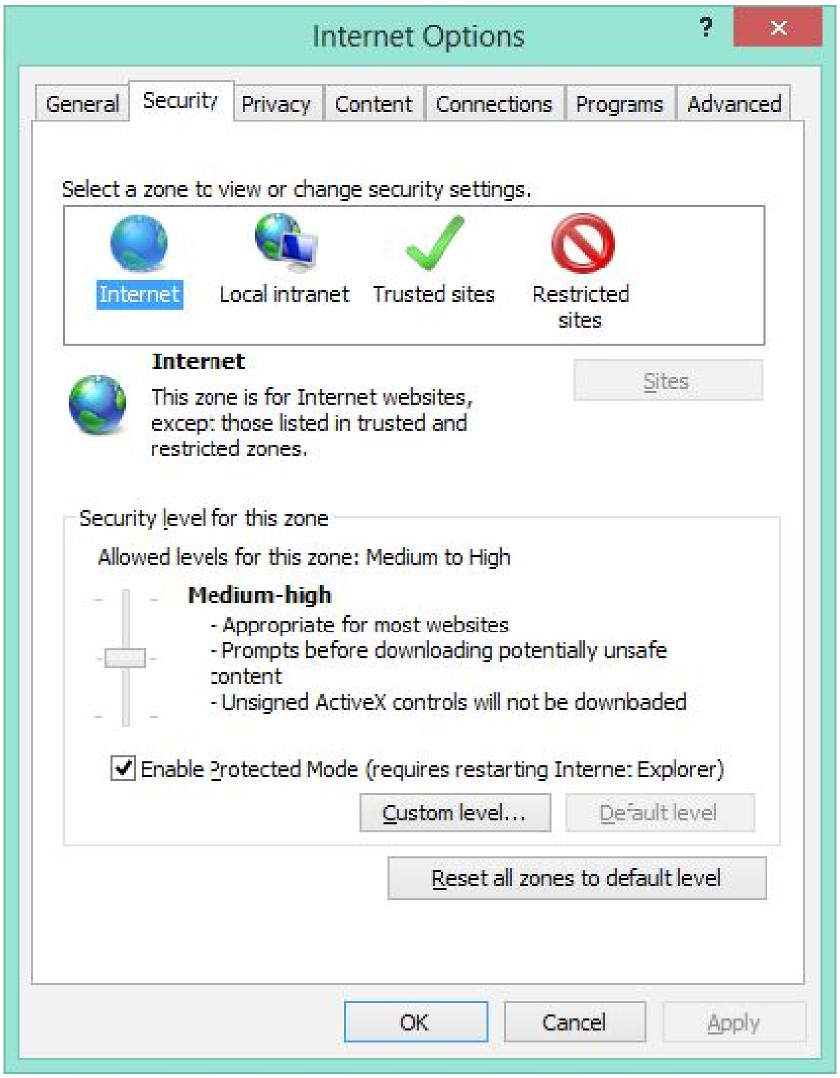
□2□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□3□□IE 7□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

① □□□□□□□□Internet□□□

② □**Internet**□□□□□□□□□□□□□□□

③ [illegible]“[illegible]”[illegible]


Internet Options
General
Security
Privacy
Content
Connections
Programs
Advanced
Select a zone to view or change security settings.
Internet
Local intranet
Trusted sites
Restricted sites
Internet
This zone is for Internet websites, except those listed in trusted and restricted zones.
Sites
Security level for this zone
Allowed levels for this zone: Medium to High
Medium-high
- Appropriate for most websites
- Prompts before downloading potentially unsafe content
- Unsigned ActiveX controls will not be downloaded
Enable Protected Mode (requires restarting Internet Explorer)
Custom level...
Default level
Reset all zones to default level
OK
Cancel
Apply

> □□□InternetExplorerDriver□□□□□□□□□□□□□□□□□□□100%□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

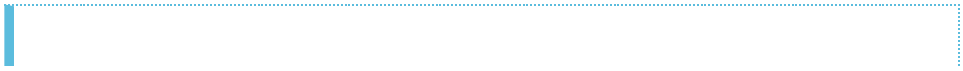
□4□□□□□□□□□□IE□□□□□□□□□□□□□□□□□□IE□□Firefox□□□

```
import os
from selenium import webdriver

# get the path of IEDriverServer
dir = os.path.dirname(__file__)
ie_driver_path = dir + "\IEDriverServer.exe"

# create a new Internet Explorer session
driver = webdriver.Ie(ie_driver_path)
driver.implicitly_wait(30)
driver.maximize_window()

# navigate to the application home page
driver.get("http://demo.magentocommerce.com/")

# get the search textbox
search_field = driver.find_element_by_name("q")
search_field.clear()

# enter search keyword and submit
search_field.send_keys("phones")
search_field.submit()

# get all the anchor elements which have product
```

```
names displayed
# currently on result page using
find_elements_by_xpath method
products =
driver.find_elements_by_xpath("//h2[@class='produc
t-name']/a")

# get the number of anchor elements found
print "Found " + str(len(products)) + " products:"

# iterate through each anchor element and print
the text that is
# name of the product
for product in products:
    print product.text

# close the browser window
driver.quit()
```

□□□□□□□□□□IE□□□□□□□□□□□□□ InternetExplorerDriver□□□□

□5□□□□□□□Selenium□□□ InternetExplorerDriver□□□□□□□□□□□□□□□□□□□ InternetExplorerDriver□□□Selenium□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□Firefox□□□□□□□ □□



□
https://code.google.com/p/selenium/wiki/

InternetExplorerDriver□□□□□□□□□IE□□□□□□

□
https://code.google.com/p/selenium/wiki/

DesiredCapabilities □□ DesiredCapabilities□□□□

1.3.2 □□Google Chrome□□□

□Google Chrome□□□□□□□□□□□□□□□□□IE□□□□□□□□□□□□□□ChromeDriver□□□ ChromeDriver□□□□□□Chromium team□□□□□

□□□□□□□□□Windows□□□□□Linux□□□□□Mac□□□□□□□□□□□□□□□□ChromeDriver□□□

□1□□ http://chromedriver.storage.googleapis.com/index.html□□ChromeDriver□□□

□2□□□□ChromeDriver□□□□□□□□□□□□□□□□□□□□□□□□□□

□3□□□□□□□□□Chrome□□□□□□□□□□□□□□□□Chrome□□□□□□□□Firefox□□□□□□

```
import os
from selenium import webdriver

# get the path of chromedriver
dir = os.path.dirname(__file__)
chrome_driver_path = dir + "\chromedriver.exe"
# remove the .exe extension on linux or mac
platform

# create a new Chrome session
driver = webdriver.Chrome(chrome_driver_path)
driver.implicitly_wait(30)
driver.maximize_window()
```

```
# navigate to the application home page
driver.get("http://demo.magentocommerce.com/")

# get the search textbox
search_field = driver.find_element_by_name("q")
search_field.clear()

# enter search keyword and submit
search_field.send_keys("phones")
search_field.submit()

# get all the anchor elements which have product
names displayed
# currently on result page using
find_elements_by_xpath method
products =
driver.find_elements_by_xpath("//h2[@class='produc
t-name']/a")

# get the number of anchor elements found
print "Found " + str(len(products)) + " products:
"

# iterate through each anchor element and print
the text that is
# name of the product
for product in products:
    print product.text

# close the browser window
driver.quit()
```

□□□□□□□□□□Chrome□□□□□□□□□□□□

ChromeDriver□□□□

□4□□□□□□□Selenium□□□ChromeDriver□
□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□Firefox□□□□□□□□□

□□□□□□□ChromeDriver□□□□
https://code.google.com/p/selenium/
wiki/ChromeDriver□
https://sites.google.com/a/chromium.or
g/chromedriver/ home□

1.4 □□□□

□□□□□□□□□□Selenium□□□□□□□□pip□□□□
□Selenium□□□□□□□□□□□□□□Selenium□Python
□□□□□□□IDE□□□□□□PyCharm□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□Firefox□□□□□□□□□□□□□□

[illegible]IE[illegible]Chrome[illegible]Selenium WebDriver[illegible]

[illegible]Selenium WebDriver[illegible]unittest[illegible]

第2章 使用unittest组织测试用例

Selenium WebDriver没有提供组织测试用例的API，因此，要执行多个测试用例，就需要借助其他的API。本章将介绍Web测试框架Selenium WebDriver与单元测试框架unittest的结合使用，以Python为例，介绍Selenium WebDriver的用法。

本章主要内容

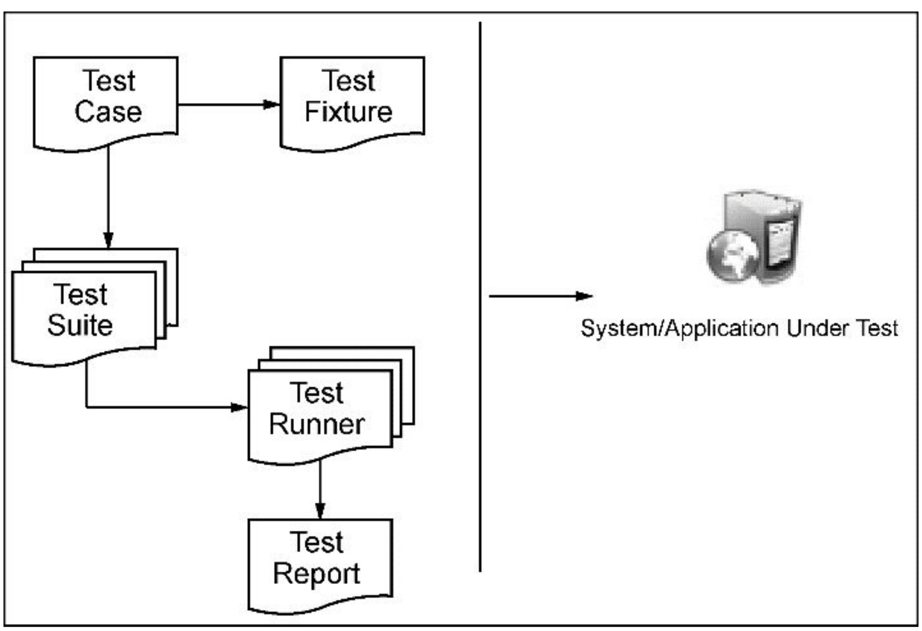
- 认识unittest；
- 使用unittest与Selenium WebDriver结合；
- 用TestCase组织测试用例；
- 使用unittest提供的多种assert方法；
- 使用批量执行TestSuite；
- 使用unittest生成测试HTML报告的方法。

2.1 unittest单元测试框架

unittest是单元测试框架PyUnit移植Java语言的单元测试框架JUnit的产物，用于对最小的软件单元进行测试验证。unittest是Python语言中自带的单元测试框架，可以直接使用。要使用unittest，必须先掌握unittest的测试固件、测试用例、测试套件和测试执行。unittest的官方网址为http://docs.python.org/2/library/unittest.html。

unittest[illegible]

- **Test Fixture**[illegible]
- **Test Case**[illegible]unittest[illegible]unittest[illegible]assert[illegible]unittest[illegible]TestCase[illegible]
- **Test Suite**[illegible]
- **Test Runner**[illegible]
- **Test Report**[illegible]



□□□unittest□□□xUnite□□□□□□□□□□□□□3□□□□3A’s□□□□□□

- **Arrange□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□**
- **Act□□□□□□□□□□**
- **Assert□□□□□□□□□□□□□□□□□□□**

□□□□□□□□□□□□□□□□□□□□□unittest□□□□□

> □□□□□□□□□□□□□□unittest□□□□□□□□Selenium WebDriver □□□□□□□Python□□□□□□□□□□□□□□□□□□□□

> - **Nose**□□□□□□□unittest□□Nose□□□□□□ https://nose.readthedocs.org/en/latest/
> - **Pytest**□Pytest□□□□□□□□□□□□□□□□□□Python□□□□□□□□□□□□□□□□□□□□□□□□□□□Pytest□□□□□□ http://pytest.org/latest/

2.1.1 TestCase□

□□□□□□□□TestCase□□TestCase□□□assert□□□□□□□□□□assert□□□□□□□□□□□□□□□□□assertEqual()□□□□□□□□□□□assertTrue() □□□□□□□□□□assertRaises() □□□□□□□□□

□□□□□□□□□□□□□□□□□□——setUp()□□□tearDown()□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□unittest□□□□□□□TestCase□□□□□□□□□□□□□□□□1□□□□Python□Selenium WebDriver□□□□□□□□□□□□□□□□□□□

□□□□□□□unittest□□□□□□□□□□□□□TestCase □□□□□□□□□□

```
import unittest
from selenium import webdriver

class SearchTest (unittest.TestCase)□
```

2.1.1.1 setUp()□□

□□□□□□□□setUp()□□□URL□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□setUp()□□□setUp()□□□□□Firefox□□□□□□properties□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
import unittest
from selenium import webdriver

class SearchTests(unittest.TestCase):
    def setUp(self):
```

```
        # create a new Firefox session
        self.driver = webdriver.Firefox()
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

        # navigate to the application home page
        self.driver.get("http://demo.magentocommerce.com/")
```

2.1.1.2 [illegible]

[illegible]setUp()[illegible]setUp()[illegible]test[illegible]TestCase[illegible]test[illegible]test runner[illegible]

[illegible]test runner[illegible]setUp()[illegible]assertEqual()[illegible]

[illegible]test_search_by_category()[illegible]

```
import unittest
from selenium import webdriver

class SearchTests(unittest.TestCase):
    def setUp(self):
        # create a new Firefox session
        self.driver = webdriver.Firefox()
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

        # navigate to the application home page
        self.driver.get("http://demo.magentocommerce.com/")

    def test_search_by_category(self):
        # get the search textbox
```

```
        self.search_field = self.driver.find_element_by_name("q")
        self.search_field.clear()

        # enter search keyword and submit
        self.search_field.send_keys("phones")
        self.search_field.submit()

        # get all the anchor elements which have product names
        # displayed currently on result page using
        # find_elements_by_xpath method
        products = self.driver.find_elements_by_xpath
          ("//h2[@class='product-name']/a")
        self.assertEqual(2, len(products))
```

2.1.1.3 □□□□

□□□setUp()□□□□□□□□□□□□□□□□□□TestCase□□□□□□□□□□□□□□□□ tearDown()□□□□□□□□□□□□□□□□□□□□□□□□□□□□□setUp()□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□setUp()□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□Firefox□□□□□□□□□tearDown()□□□□□Firefox□□□ □□□□□□□

```
import unittest
from selenium import webdriver

class SearchTests(unittest.TestCase):
    def setUp(self):
        # create a new Firefox session
        self.driver = webdriver.Firefox()
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

        # navigate to the application home page
        self.driver.get("http://demo.magentocommerce.com/")

    def test_search_by_category(self):
        # get the search textbox
        self.search_field = self.driver.find_element_by_name("q")
        self.search_field.clear()

        # enter search keyword and submit
        self.search_field.send_keys("phones")
        self.search_field.submit()
```

```
        # get all the anchor elements which have product names
        # displayed currently on result page using
        # find_elements_by_xpath method
        products = self.driver.find_elements_by_xpath
          ("//h2[@class='product-name']/a")
        self.assertEqual(2, len(products))

    def tearDown(self):
        # close the browser window
        self.driver.quit()
```

2.1.1.4 [illegible]

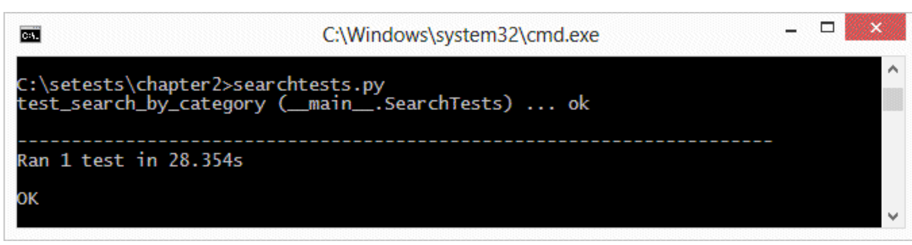
[illegible]main[illegible]verbosity[illegible]

```
if __name__ == '__main__':
    unittest.main(verbosity=2)
```

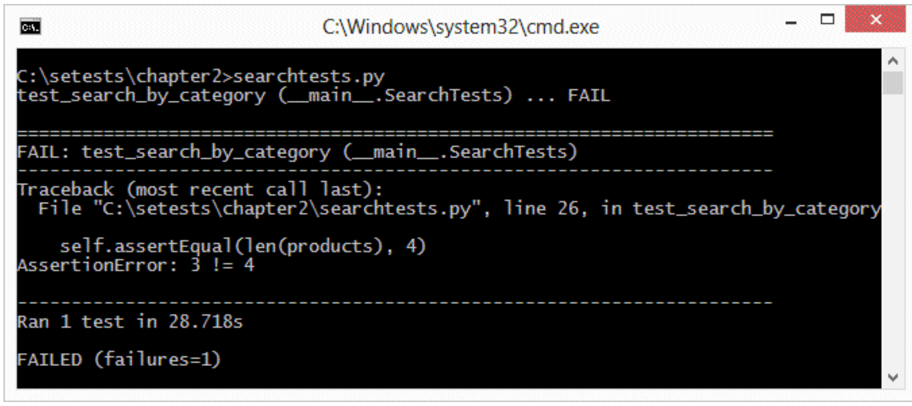
[illegible]Python[illegible]searchtests.py[illegible]

```
python searchtests.py
```

[illegible]unittest[illegible]



[illegible]



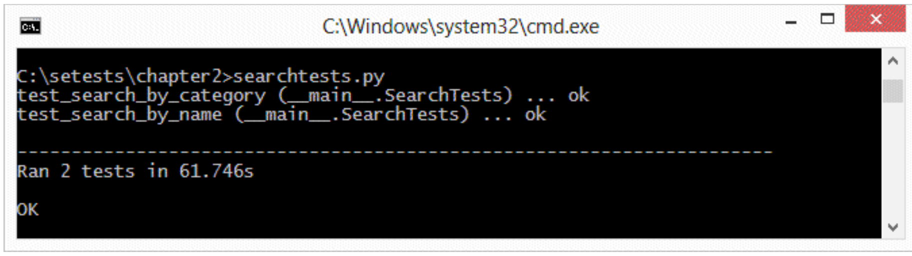
[illegible]AssertionError[illegible]

2.1.1.5 [illegible]

[illegible]test[illegible]

```
def test_search_by_name(self):
        # get the search textbox
        self.search_field = self.driver.find_element_by_name("q")
        self.search_field.clear()

        # enter search keyword and submit
        self.search_field.send_keys("salt shaker")
        self.search_field.submit()

        # get all the anchor elements which have
        # product names displayed
        # currently on result page using
        # find_elements_by_xpath method
        products = self.driver.find_elements_by_xpath
          ("//h2[@class='product-name']/a")
        self.assertEqual(1, len(products))
```

[illegible]Firefox[illegible]setUp()[illegible]tearDown()[illegible]



2.1.2 □□□□setUp()□□□tearDown()□□

□□□□□□□□□□□□setUp()□□□□□□□□□□□□□□□Firefox□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Firefox□□□□□□□□□□□□□□□□□□□□□□□□□□setUpClass()□□□tearDownClass()□□□@classmethod□□setUpClass()□□□tearDownClass()□□□□□□@classmethod□□□

```
import unittest
from selenium import webdriver

class SearchTests(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        # create a new Firefox session
        cls.driver = webdriver.Firefox()
        cls.driver.implicitly_wait(30)
        cls.driver.maximize_window()

        # navigate to the application home page
        cls.driver.get("http://demo.magentocommerce.com/")
        cls.driver.title

    def test_search_by_category(self):
        # get the search textbox
        self.search_field = self.driver.find_element_by_name("q")
        self.search_field.clear()

        # enter search keyword and submit
        self.search_field.send_keys("phones")
        self.search_field.submit()
```

```
        # get all the anchor elements which have product names
        # displayed currently on result page using
        # find_elements_by_xpath method
        products = self.driver.find_elements_by_xpath
          ("//h2[@class='product-name']/a")
        self.assertEqual(2, len(products))

    def test_search_by_name(self):
        # get the search textbox
        self.search_field = self.driver.find_element_by_name("q")
        self.search_field.clear()

        # enter search keyword and submit
        self.search_field.send_keys("salt shaker")
        self.search_field.submit()

        # get all the anchor elements which have product names
        # displayed currently on result page using
        # find_elements_by_xpath method
        products = self.driver.find_elements_by_xpath
          ("//h2[@class='product-name']/a")
        self.assertEqual(1, len(products))

    @classmethod
    def tearDownClass(cls):
        # close the browser window
        cls.driver.quit()

    if __name__ == '__main__':
        unittest.main()
```

□□□□□□□□□□□□□□□Firefox□□□□□□□□□□□□□□□□□

□□□□□□□□@classmethod□□□□□□□□□□
https://docs.python.org/2/library/
functions.html#classmethod□

2.1.3 □□

unittest[illegible]TestCase[illegible]3[illegible]

unittest[illegible]xUnit [illegible]

[illegible]	[illegible]	[illegible]
assertEqual(a, b [,msg])	a == b	[illegible]a[illegible]b[illegible]msg[illegible] [illegible] assertEqual(element.text, "10")
assertNotEqual(a, b[,msg])	a != b	
assertTrue(x[,msg]))	bool(x) is True	[illegible]True[illegible]False[illegible] [illegible] assertTrue(element.is_displayed())
assertFalse(x[,msg]))	bool(x) is False	
assertIsNot(a, b[,msg]))	a is not b	
assertRaises(exc, fun, *args, **kwds)	fun(*args, **kwds) raises exc	[illegible] NoSuchElementFoundexception
assertRaisesRegexp(exc, r, fun, *args, **kwds)	fun(*args,**kwds) raises exc and the message matches regex r	
assertAlmostEqual(a, b)	round(a-b, 7) == 0	[illegible]
assertNotAlmostEqual(a, b)	round(a-b, 7) != 0	
assertGreater(a, b)	a > b	[illegible]assertEqual()[illegible]
assertGreaterEqual(a, b)	a >= b	
assertLess(a, b)	a < b	
assertLessEqual(a, b)	a <= b	
assertRegexpMatches(s, r)	r.search(s)	[illegible]
assertNotRegexpMatches(s, r)	not r.search(s)	
assertMultiLineEqual(a, b)	strings	[illegible]assertEqual()[illegible]

		[illegible]
assertListEqual(a, b)	lists	[illegible]list[illegible]
fail()		[illegible]assert[illegible]

2.1.4 [illegible]

[illegible]unittest[illegible]TestSuites[illegible]TestSuites[illegible]TestLoader[illegible]TestRunner[illegible]

[illegible]TestSuites[illegible]

```
import unittest
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.by import By
from __builtin__ import classmethod

class HomePageTest(unittest.TestCase):
    @classmethod
    def setUp(cls):
        # create a new Firefox session """
        cls.driver = webdriver.Firefox()
        cls.driver.implicitly_wait(30)
        cls.driver.maximize_window()
        # navigate to the application home page """
        cls.driver.get("http://demo.magentocommerce.com/")

    def test_search_field(self):
        # check search field exists on Home page
        self.assertTrue(self.is_element_present(By.NAME,"q"))

    def test_language_option(self):
        # check language options dropdown on Home page
        self.assertTrue(self.is_element_present
          (By.ID,"select-language"))

    def test_shopping_cart_empty_message(self):
        # check content of My Shopping Cart block on Home page
```

```
        shopping_cart_icon = \
            self.driver.find_element_by_css_selector
              ("div.header-minicart span.icon")
        shopping_cart_icon.click()

        shopping_cart_status = \
             self.driver.find_element_by_css_selector
               ("p.empty").text
        self.assertEqual("You have no items in your shopping cart.",
shopping_cart_status)

        close_button =  self.driver.find_element_by_css_selector
          ("div.minicart-wrapper a.close")
        close_button.click()

    @classmethod
    def tearDown(cls):
        # close the browser window
        cls.driver.quit()

    def is_element_present(self, how, what):
        """
        Utility method to check presence of an element on page
        :params how: By locator type
        :params what: locator value
        """
        try: self.driver.find_element(by=how, value=what)
        except NoSuchElementException, e: return False
        return True

    if __name__ == '__main__':
        unittest.main(verbosity=2)
```

□□□□TestSuite□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□TestSuite□□□□□□□□TestLoader□TextTestRunner□□□□□□□□□□□□□□□□

```
import unittest
from searchtests import SearchTests
from homepagetests import HomePageTest

# get all tests from SearchProductTest and HomePageTest class
search_tests = unittest.TestLoader().loadTestsFromTestCase
(SearchTests)
home_page_tests = unittest.TestLoader().loadTestsFromTestCase
(HomePageTest)
```

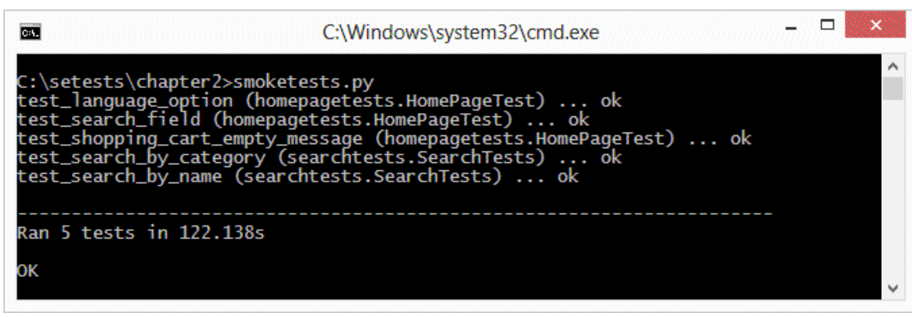
```
# create a test suite combining search_test and home_page_test
smoke_tests = unittest.TestSuite([home_page_tests, search_tests])

# run the suite
unittest.TextTestRunner(verbosity=2).run(smoke_tests)
```

□□TestLoader□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□TestRunner□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□

```
python smoketests.py
```

□□□□SearchProductTest□□HomePageTest□□□□□□□□□□□□□□□□□□□□□□□□□□□□



2.2 □□HTML□□□□□□□

unittest□□□unittest□□□□□□□□□□□□□□□□□□□□□□□□□□□Wai Yip Tung□□□unittest□□□HTMLTestRunner□□□□

□□□□□□□□□□□□HTMLTestRunner□□□□□□□□□□□
https://pypi.python.org/pypi/HTMLTestRunner□

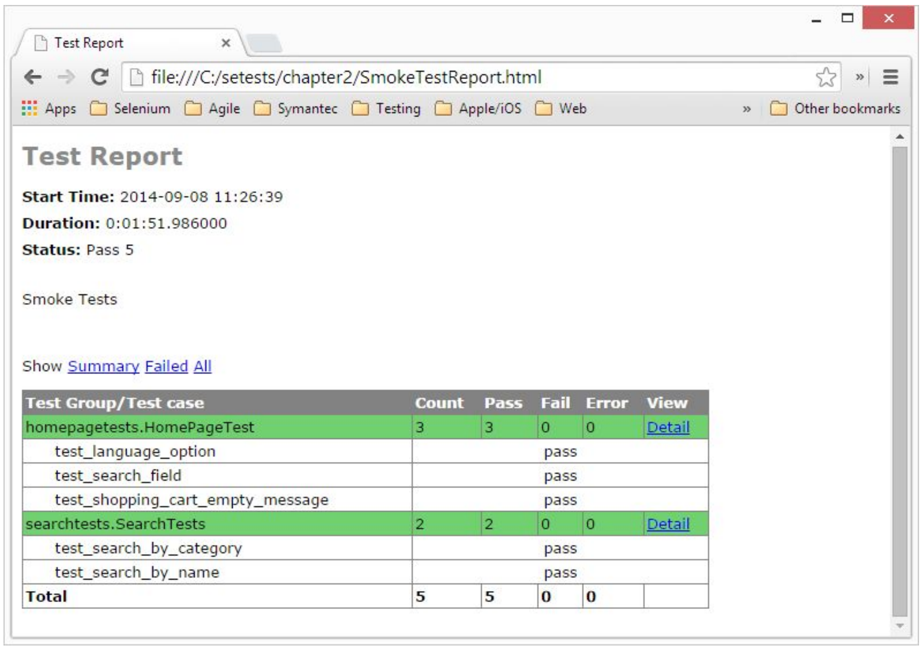
> HTMLTestRunner□□□□□□□□□□□□□□□□

□□□□□□□□HTMLTestRunner□□□□□□□□□□□□□□□□□□□□
□□□□□□□□HTMLTestRunner□□□□□□□□□□□□□□□□□□□□□□□
□□□HTMLTestRunner□□□□□□□□□□□

```
import unittest
import HTMLTestRunner
import os
from searchtests import SearchTests
from homepagetests import HomePageTest

# get the directory path to output report file
dir = os.getcwd()

# get all tests from SearchProductTest and HomePageTest class
search_tests =
unittest.TestLoader().loadTestsFromTestCase(SearchTests)
home_page_tests =
unittest.TestLoader().loadTestsFromTestCase(HomePageTest)

# create a test suite combining search_test and home_page_test
smoke_tests = unittest.TestSuite([home_page_tests, search_tests])

# open the report file
outfile = open(dir + "\SmokeTestReport.html", "w")

# configure HTMLTestRunner options
runner = HTMLTestRunner.HTMLTestRunner(
                 stream=outfile,
                 title='Test Report',
                 description='Smoke Tests'
                 )
```

```
# run the suite using HTMLTestRunner
runner.run(smoke_tests)
```

□□□□□□□HTMLTestRunner□unittest□□□



2.3 □□□□

□□□□□□□□□□unittest□□□□□□□Selenium WebDriver□□□□□□□□□□□setUp()□□□tearDown()□□□TestCase□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□unittest□□□HTMLTestRunner□□□□□□□HTML□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□HTML□□□□□□

第3章　要素の操作

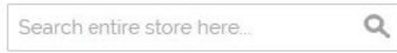
Webページは、基本的に**HTML**で構造を定義し、**CSS**やJavaScriptで装飾や動作を定義します。Webページを操作するためには、**URL**でWebページを開いた後、Webページの構造を定義しているHTMLや、装飾やJavaScript、CSSなどを取得する必要があります。Webページを操作するためには、Webページの構造を定義しているHTMLを取得する必要があります。Seleniumでは、**WebElements**と

この章で学ぶこと

- 基本的なSelenium WebDriverの操作方法
- 要素を取得するための様々な方法
- 要素を取得するためにID、name、class、タグ名、XPath、CSSを使う方法
- 要素をfind_element_byで取得するためのSelenium WebDriverの仕組み

Seleniumは、Seleniumを使うためのライブラリであり、Seleniumを使うためのライブラリであり、Seleniumを使うためのライブラリです。

Webページは、Webページを構成するHTML、CSS、JavaScriptなどのファイルを取得し、tagなどを使って、Webページを構成するHTMLを取得します。



□□□□□□□□□□HTML□□□

```
<form id="search_mini_form" action=
  "http://demo.magentocommerce.com/catalogsearch/result/"
  method="get">
     <div class="form-search">
        <label for="search">Search:</label>
        <input id="search" type="text" name="q" value=""
          class="input-text" maxlength="128" />
        <button type="submit" title="Search"
          class="button"><span><span>Search</span></span></button>
        <div id="search_autocomplete" class="search-
          autocomplete"></div>
        <script type="text/javascript">
        //<![CDATA[
            var searchForm = new Varien.searchForm
              ('search_mini_form', 'search', 'Search entire store
              here...');
            searchForm.initAutocomplete
              ('http://demo.magentocommerce.com
              /catalogsearch/ajax/suggest/',
              'search_autocomplete');
        //]]>
        </script>
     </div>
</form>
```

□□□□□□□□□□□□□□□□□□□□□□□□□□<form>□□□□<input>□□□□□□□□□□□<label>□□□□□□□□□JavaScript□□□□□<script>□□□□

□□□□□<input>□□□□□id□type□name□value□class□maxlength□□□□□□

```
<input id="search" type="text" name="q" value=""
   class="input-text" maxlength="128" />
```

□□□HTML□□□JavaScript□□□

□□□□□□HTML□CSS□JavaScript□□□□□□□□□□□□□□□□□□□□□□□□□□
WebDriver□□□□□□□□□

3.1 □□□□□□□□□□□□□

□□□Selenium□□DOM□□□JavaScript□□□□CSS□□□□□□□□□□□□□□□□□□□□□□□□□□□□

3.1.1 □□□□□□Firebug□□□□□□□□

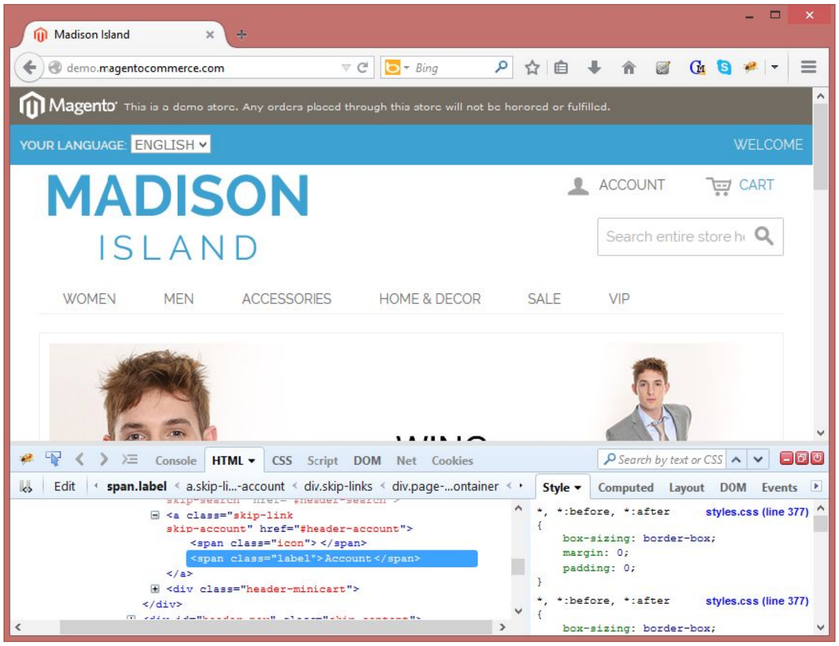
□□Firebug□□□

□1□□□□□□□□□□□□□□□□Firebug□□□

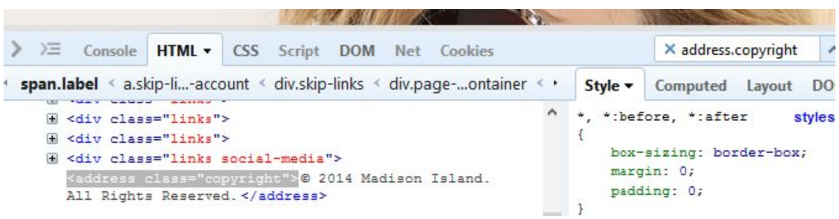
https://addons.mozilla.org/en-us/firefox/addon/firebug/

□2□□□□Firebug□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□3□□□□**Firebug**□□□□□□□□□□□□□□□□□HTML□□□□□□□□□□□□□□□□□□□□□□



□4□□□□□□□□Firebug□XPath□CSS□□□□□□Firebug□□□

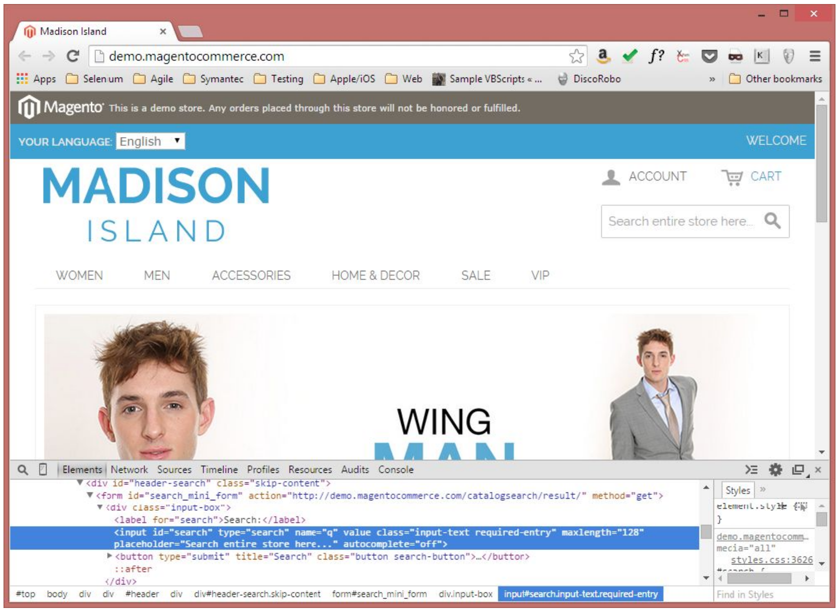


3.1.2 □□□Chrome□□□□□□□□□□

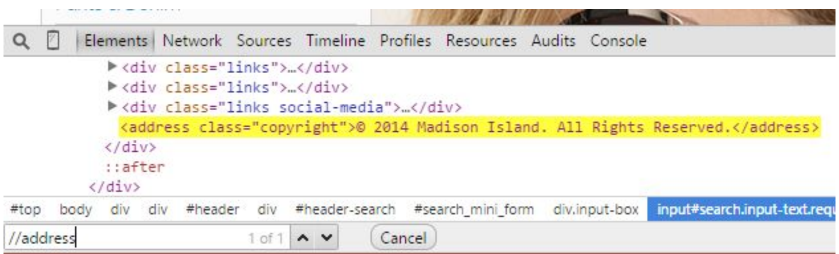
□□Chrome□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□1□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□**N**□□□□

□□□□□□□□□□□□□Firebug□□□□□□□□□□□□□□□



（2□□□□□Firebug□□□□□□□□XPath□CSS□□□□□□□□□□□□□Elements□□□□□Ctrl+F□□□□□□□□□□□□□□XPath□CSS□□□□Firebug□□□□□□□□□□□□□□□□□□□□

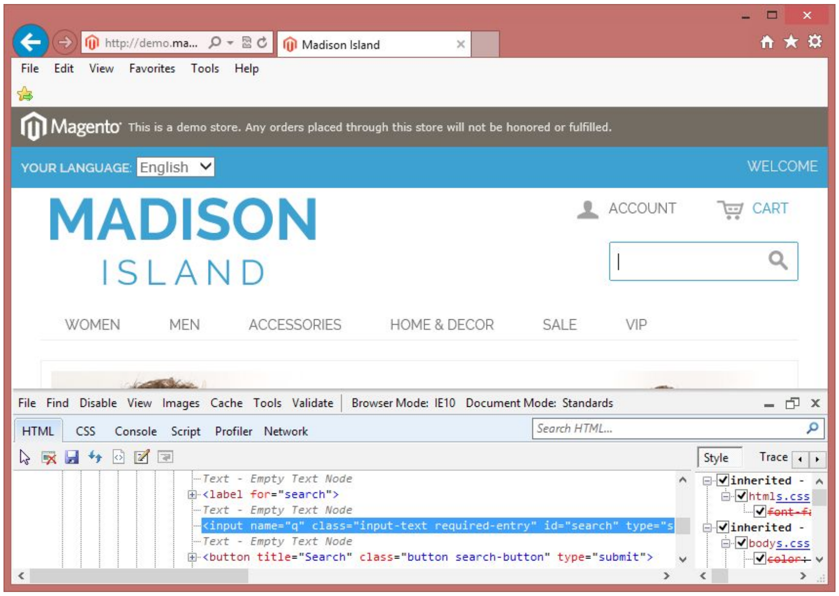


3.1.3 □IE□□□□□□□□□

□□□□□IE□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□1□□F12□□□□□□□□□□□□□□□□□□

□2□□□□□□□□□□□“□□”□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□HTML□□□□□□□□□□



□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□JavaScript□□□

3.2 □□□□

□□□□□□Selenium□□

□□□□□□□□□□□□□□□□□□□Selenium□□□□□□□□□□□□Selenium□□Selenium□□□□□□□□□□□□□□□□□□□□□□□□□□

Selenium□□□□find_element_by □□WebElement□□□□□□□□□□□□ NoSuchElementException□□□□□□□Selenium□□□□□ find_elements_by □□□□□□□□□□□□□□□□□□□□□□□□□□□□□list□□□□□□□

Selenium[illegible]8[illegible]find_element_by [illegible]

[illegible]

[illegible]	[illegible]	[illegible]	[illegible]
find_element_by_id(id)	[illegible]ID[illegible]	id[illegible]ID	driver.find_element_by_id
find_element_by_name(name)	[illegible] name [illegible]	name[illegible] name	driver.find_element_by_na
find_element_by_class_name(name)	[illegible] class [illegible]	name[illegible]	driver.find_element_by_cl
find_element_by_tag_name(name)	[illegible]tag name [illegible]	name[illegible]tag name	driver.find_element_by_ta
find_element_by_xpath(xpath)	[illegible] XPath [illegible]	XPath[illegible] XPath	driver.find_element_by_xp

□ □	□ □	□ □	□
find_element_by_css_selector(css_selector)	□□ CSS□ □□□□ □□□	css_selector□ □□□CSS□□□	driver.find_element_by_cs
find_element_by_link_text(link_text)	□□□□ □□□□ □□□□ □□□□ □□□	link_text□□□□ □	driver.find_element_by_lir
find_element_by_partial_link_text(link_text)	□□□□ □□□□ □□□□ □□□□ □□□□ □	link_text□□□□ □□□	driver.find_element_by_pa

find_elements_by □□□□□□□□□□□□□□□□□□□□□□□□□□□

□ □	□ □	□ □	
find_elements_by_id(id_)	□□□□ □ID□□ □□□□ □□□□	id_□□□□ID	driver.find_elements_by_

□ □	□ □	□ □	
find_elements_by_name(name)	□□□□ □ name □□□□ □□□□ □□	name□□□□ name	driver.find_elements_by_
find_elements_by_class_name(name)	□□□□ □ class □□□□ □□□□	name□□□□□□	driver.find_elements_by_
find_elements_by_tag_name(name)	□□□□ □tag name □□□□ □□□	name□tag name	driver.find_elements_by_
find_elements_by_xpath(xpath)	□□ XPath □□□□ □□□	XPath□□□□ XPath	driver.find_elements_by_
find_elements_by_css_selector(css_selector)	□□ CSS□ □□□□ □□□□ □	css_selector□ □□□CSS□□□	driver.find_elements_by_

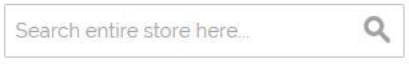
□ □	□ □	□ □	
find_elements_by_link_text(text)	□□□□ □□□□ □□□□ □□□□ □□□□ □	text□□□□□	driver.find_elements_by_
find_elements_by_partial_link_text(link_text)	□□□□ □□□□ □□□□ □□□□ □□□□ □□□	link_text□□□□ □□□	driver.find_elements_by_

3.2.1 ID□□

□□ID□□□□□□□□□□□□□□□□□□□□□□find_element_by_id()□ find_elements_by_id()□□□□□ID□□□□□□□□□□□□□□□□□

find_element_by_id()□□□□□□□ID□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □NoSuchElementException□□□

□□□□□□□□□□□□□□□□□□



□□□□HTML□□□□□□□□□□□□ID□□□□□search□

```
<input id="search" type="text" name="q" value=""
  class="input-text" maxlength="128" autocomplete="off">
```

□□□□□□□find_element_by_id() □□□id□□search□□□□□□□□□□□□□ maxlength□□□□□

```
def test_search_text_field_max_length(self):
  # get the search textbox
  search_field = self.driver.find_element_by_id("search")

  # check maxlength attribute is set to 128
  self.assertEqual("128", search_field.get_attribute("maxlength"))
```

□□□□□□□find_elements_by_id() □□□□□□□□□□ID□□□□□□□□

3.2.2 name□□

□□name□□□□□□□□□□□□□□□□□□□□find_element_by_name() □ find_elements_by_name()□□□□□□□□name□□□□□□□□□□□□□□□□name □□□□□□□□□□□□□□□□□□□□□□NoSuchElementException□□□□

□□□□□□□□□□□□□□□□name□□□□□□□□□ID□□□□□□□□□□□□□□□□□

```
# get the search textbox
self.search_field = self.driver.find_element_by_name("q")
```

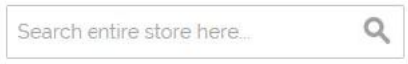
□□□□□□□find_elements_by_name() □□□□□□□□□□name□□□□□□□

3.2.3 class□□

□□□□ID□name□□□□□□□□□class□□□□□□□□class□□□□CSS□□□□□ □□find_element_by_class_name()□ find_elements_by_class_name()□□□□□□□□class□□□□□□□□□□□□□□ □□class□□□□□□□□□□□□□□□□□□□□□□□□NoSuchElementException□□□□

> □□□□□ID□name□class□□□□□□□□□□□□□□□□□□□□□□□□Selenium WebDriver□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□find_element_by_class_name() □□□□□□□□□



□□□□□□□□□□□□□HTML□□□<button>□□□□□□□□□□□□class□□□□□□□□□□□□□□□

```
<button type="submit" title="Search"
  class="button"><span><span>Search</span></span></button>
```

□□□□□class□□□□□□□□□□□□□□□□□

```
def test_search_button_enabled(self):
    # get Search button
    search_button = self.driver.find_element_by_class_name
      ("button")

    # check Search button is enabled
    self.assertTrue(search_button.is_enabled())
```

□□□□□□□□find_elements_by_class_name() □□□□□□□□□□□class□□□□□□□□□

3.2.4 tag□□

find_element_by_tag_name()□find_elements_by_tag_name() □□□□□□HTML□□□tag name□□□□□□□□□□□□□□□□□JavaScript□□DOM□□getElementsBy TagName()□□□□□□□□□□□□□tag name□□□□□□□□□□□□□□□□NoSuchElementException□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□<tr> □tag name□□□□□□□
table□□□□□□□□□□□□

□□□□□□□□banner□□□□□□HTML□□□□□□□□□□□□<img>□□<ul>□□
□□

□□□banner□□□□□□□□□<ul>□□□□□□<img>□□□□

```
<ul class="promos">
   <li>
      <a href="http://demo.magentocommerce.com/home-decor.html">
         <img src="/media/wysiwyg/homepage-three-column-promo-
           01B.png" alt="Physical & amp; Virtual Gift Cards">
      </a>
   </li>
   <li>
      <a href="http://demo.magentocommerce.com/vip.html">
         <img src="/media/wysiwyg/homepage-three-column-promo-
           02.png" alt="Shop Private Sales - Members Only">
      </a>
   </li>
   <li>
      <a href="http://demo.magentocommerce.com/accessories/ bags-
luggage.html">
         <img src="/media/wysiwyg/homepage-three-column-
           promo-03.png" alt="Travel Gear for Every Occasion">
      </a>
   </li>
</ul>
```

□□□□find_elements_by_tag_name()□□□□□□□□banner□□□□□□□□
find_element_by_class_name()□□□□□□□banner□□□□□
find_elements_by_tag_name()□□□□□<img>□tag name□□□□□□□□□
banners□□□

```
def test_count_of_promo_banners_images(self):
   # get promo banner list
   banner_list = self.driver.find_element_by_class_name("promos")
```

```
    # get images from the banner_list
    banners = banner_list.find_elements_by_tag_name("img")

    # check there are 20 tags displayed on the page
    self.assertEqual(2, len(banners))
```

3.2.5 XPath□□

XPath□□□XML□□□□□□□□□□□□□□□□□□□□□□XPath□□□Selenium□□□□XPath□□□Web□□□□□□

□□□□□□ID□name□class□□□□□□□□□□□□□□XPath□□□□□□□□□□□□□□□□□□□□□□□□ID□name□□□□□□□□□□□□□□□□□□□□□□starts-with()□contains()□ends-with()□□□□□□□

□□□□□XPath□□□□□□□□□□□□

□□□□□□XPath□□□□□□□□□Selenium Testing Tools Cookbook□□Packt Publishing□

□□□□□□□□find_element_by_xpath()□find_elements_by_xpath() □□□□□□□□□□□□□□□□□banner□□□banner□□□□□□□□

□□□□□“Shop Private Sales”□banner□□□<img> □tag□□□□□□□ID□name□class□□□□□□□□□□□□□□□<img>□□□□□□□□□□□

□find*by* tag_name()□□□□□□

```
<ul class="promos">
    ...
    <li>
       <a href="http://demo.magentocommerce.com/vip.html">
          <img src="/media/wysiwyg/homepage-three-column-
            promo-02.png" alt="Shop Private Sales - Members Only">
       </a>
    </li>
    ...
</ul>
```

□□□□□find_element_by_xpath()□□□<img>□□□ alt □□□□□□
□□□□□□

□□□□

```
def test_vip_promo(self):
    # get vip promo image
    vip_promo = self.driver.\
      find_element_by_xpath("//img[@alt='Shop Private Sales - Members
Only']")

    # check vip promo logo is displayed on home page
    self.assertTrue(vip_promo.is_displayed())
    # click on vip promo images to open the page
    vip_promo.click()
    # check page title
    self.assertEqual("VIP", self.driver.title)
```

□□□□□□find_elements_by_xpath() □□□□□□□□XPath□□□□□□
□□

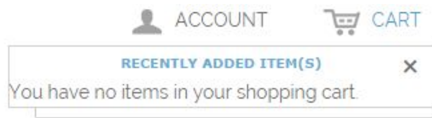
3.2.6 CSS□□□□□

CSS□□□□□□□□□□□□□□HTML□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□CSS□□□□□□□□□□□□□ID□class□type□attribute□value
□□□

[illegible]XPath[illegible]Selenium[illegible]CSS[illegible]CSS[illegible]

[illegible]find_element_by_css_selector()[illegible]find_elements_by_css_selector()[illegible]

[illegible]“[illegible]”[illegible]



HTML[illegible]

```
<div class="minicart-wrapper">
<p class="block-subtitle">
   Recently added item(s)
   <a class="close skip-link-close" href="#" title="Close">
     ×</a>
</p>
   <p class="empty">You have no items in your shopping cart.
     </p>
</div>
```

[illegible]CSS[illegible]

```
def test_shopping_cart_status(self):
   # check content of My Shopping Cart block on Home page
   # get the Shopping cart icon and click to open the
   # Shopping Cart section
   shopping_cart_icon = self.driver.\
      find_element_by_css_selector("div.header-minicart span.icon")
   shopping_cart_icon.click()

   # get the shopping cart status
   shopping_cart_status = self.driver.\
      find_element_by_css_selector("p.empty").text
   self.assertEqual("You have no items in your shopping cart.",
shopping_cart_status)
```

```
    # close the shopping cart section
    close_button = self.driver.\
        find_element_by_css_selector("div.minicart-wrapper a.close")
    close_button.click()
```

□□□□□□□□□□□□□□□□□□□□□□□□tag□class name□□□□□□□□□□□□□□□

```
shopping_cart_icon = self.driver.\
       find_element_by_css_selector("div.header-minicart span.icon")
```

□□□□□tag □□ □□□□□□□□“**.header-minicart**”□□□□□□□□□□□□□□“.icon”□□□□

□□□□□□□□CSS□□□□□□□□□□□□Selenium Testing Tools Cookbook□□Packt Publishing□

3.2.7 Link□□

find_element_by_link_text() □ find_elements_by_link_text()□□□□□□□□□□□□□□□□□□□□□

□1□□□□□□□Account□□□□□□□□□□□□□□□□□find_element_by_link_text()□□□

□2□□□□□□HTML□□□□□□□□

```
<a href="#header-account" class="skip-link skip-account">
  <span class="icon"></span>
  <span class="label">Account</span>
</a>
```

□3□□□□□□□□□□□□□□□□Account□□□□□□□□□□□□□□□□

```
def test_my_account_link_is_displayed(self):
    # get the Account link
```

```
    account_link =
      self.driver.find_element_by_link_text("ACCOUNT")

    # check My Account link is displayed/visible in
    # the Home page footer
    self.assertTrue(account_link.is_displayed())
```

□□□□□□□find_elements_by_link_text()□□□□□□□□□□□□□□□□□□

3.2.8 Partial link□□

find_element_by_partial_link_text()□find_elements_by_partial_link_text()□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□1□□□□□□□□□□□□□□□□□□□□□Account□□□□□□□□□□□□header□□□□Account□□□□□□□□□□□□□footer□□□□My Account□□□□□

□2□□□□□find_elements_by_partial_link_text()□□□□□□□□□□□“Account”□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
def test_account_links(self):
    # get the all the links with Account text in it
    account_links = self.driver.\
        find_elements_by_partial_link_text("ACCOUNT")

    # check Account and My Account link is displayed/visible in the Home
page footer
    self.assertTrue(2, len(account_links))
```

3.3 □□□□

□□□□□□□□□□□□□□□□□□find_element_by□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□1□□□□□□□homepagetest.py□Python□□□□□□□□□□□□□□□□□□□

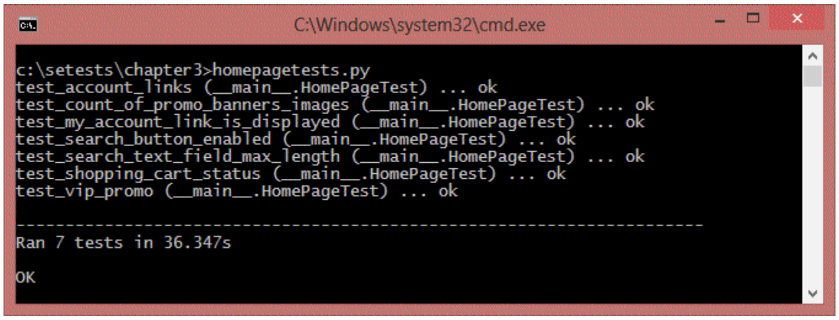
```
import unittest
from selenium import webdriver

class HomePageTest(unittest.TestCase):
   @classmethod
   def setUpClass(cls):
      # create a new Firefox session
      cls.driver = webdriver.Firefox()
      cls.driver.implicitly_wait(30)
      cls.driver.maximize_window()

      #navigate to the application home page
      cls.driver.get('http://demo.magentocommerce.com/')

   def test_search_text_field_max_length(self):
      # get the search textbox
      search_field = self.driver.
        find_element_by_id("search")

      # check maxlength attribute is set to 128
      self.assertEqual("128", search_field.get_attribute
      ("maxlength"))

   def test_search_button_enabled(self):
      # get Search button
      search_button = self.driver.
        find_element_by_class_name("button")

      # check Search button is enabled
      self.assertTrue(search_button.is_enabled())

   def test_my_account_link_is_displayed(self):
      # get the Account link
      account_link =
        self.driver.find_element_by_link_text("ACCOUNT")

      # check My Account link is displayed/visible in
      # the Home page footer
      self.assertTrue(account_link.is_displayed())

   def test_account_links(self):
      # get the all the links with Account text in it
      account_links = self.driver.\
        find_elements_by_partial_link_text("ACCOUNT")

      # check Account and My Account link is
      # displayed/visible in the Home page footer
      self.assertTrue(2, len(account_links))

   def test_count_of_promo_banners_images(self):
      # get promo banner list
      banner_list = self.driver.
        find_element_by_class_name("promos")

      # get images from the banner_list
      banners = banner_list.
```

```
        find_elements_by_tag_name("img")

      # check there are 3 banners displayed on the page
      self.assertEqual(2, len(banners))

   def test_vip_promo(self):
      # get vip promo image
      vip_promo = self.driver.\
         find_element_by_xpath("//img[@alt=
         'Shop Private Sales - Members Only']")

      # check vip promo logo is displayed on home page
      self.assertTrue(vip_promo.is_displayed())
      # click on vip promo images to open the page
      vip_promo.click()
      # check page title
      self.assertEqual("VIP", self.driver.title)

   def test_shopping_cart_status(self):
      # check content of My Shopping Cart block
      # on Home page
      # get the Shopping cart icon and click to
      # open the Shopping Cart section
      shopping_cart_icon = self.driver.\
        find_element_by_css_selector("div.header-
          minicart span.icon")
      shopping_cart_icon.click()

      # get the shopping cart status
      shopping_cart_status = self.driver.\
        find_element_by_css_selector("p.empty").text
      self.assertEqual("You have no items in your shopping
      cart.", shopping_cart_status)
      # close the shopping cart section
      close_button = self.driver.\
        find_element_by_css_selector("div.minicart-
        wrapper a.close")
      close_button.click()

   @classmethod
   def tearDownClass(cls):
      # close the browser window
      cls.driver.quit()

if __name__ == '__main__':
   unittest.main(verbosity=2)
```

□2□□□py□□□□□□□□□□□□□□□□

```
python homepagetest.py
```

□3□□□□□□□□□unittest□□□7□□□□□□□□□OK□□□□□□□□



3.4 [illegible]

[illegible]Selenium[illegible]find_element_by[illegible]ID[illegible]name[illegible]class name[illegible]tag name[illegible]XPath[illegible]CSS[illegible]

[illegible]Selenium API[illegible]

[illegible]Selenium WebDriver[illegible]JavaScript[illegible]

第4章　Selenium Python API详解

Web应用程序由HTML文档组成，浏览器解析HTML文档并呈现给用户。用户与网页交互时，浏览器会将这些操作转换为对HTML元素的操作。

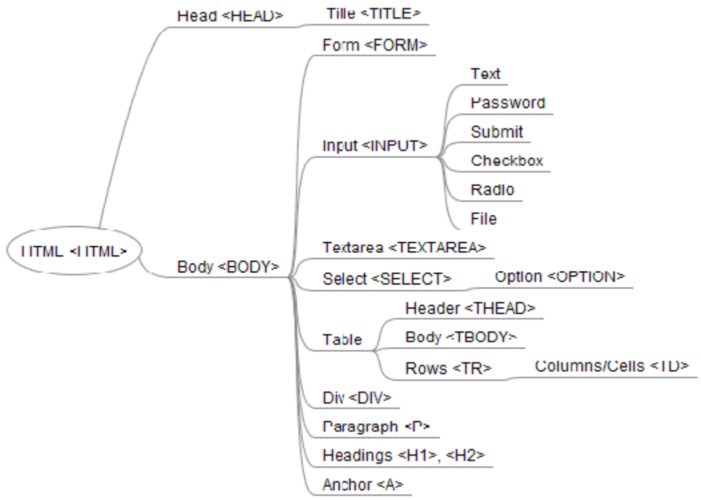
为了实现对Web应用程序的自动化测试，我们需要模拟用户的操作，例如点击按钮、输入文本、选择下拉框等。这些操作通常涉及到HTML元素，例如<form>元素用于提交表单数据，GET和POST请求用于向服务器发送数据。

本章将介绍：

- 如何使用WebDriver 和 WebElement进行交互
- 如何WebDriver和WebElement的常用方法来操作Web应用程序的元素
- 如何Select类来处理下拉框元素
- 如何JavaScript 来执行一些复杂的操作

4.1　HTML基础知识

HTML文档由一系列元素组成，这些元素通过标签来定义，例如<form>、<input>、<button>、<label>等。在Web自动化测试中，我们需要识别这些元素，并对它们进行操作。为了识别Web应用程序中的HTML元素，我们需要了解HTML文档的结构，以及如何使用选择器来定位元素。



Selenium WebDriver□“□□□□□□Web□□□□□□□□□□□□Web□□□□□□□□”□□□□□□□□□

4.2 WebDriver□□

WebDirver□□□□□□□□□□□□□□□□□□□□□□□□□□WebDirver□□cookies□□□□□□□□□□□□□□□□□□□□□□□□□□□□□WebDirver□□

□□□□□□□□□□□□WebDirver□properties□□□□□□

http://selenium.googlecode.com/git/docs/api/py/webdriver_remote/selenium.webdriver.remote.webdriver.html#moduleselenium.webdriver.remote.webdriver

4.2.1 WebDriver□□

WebDriver□□□□□□□□□□□□□□

□□/□□	□　□	□　□
current_url	□□□□□□□URL□□	driver.current_url
current_window_handle	□□□□□□□□□	driver.current_window_handle
name	□□□□□□□□□□□□□	driver.name
orientation	□□□□□□□□□	driver.orientation
page_source	□□□□□□□□□□	driver.page_source
title	□□□□□□□□□	driver.title
window_handles	□□□□session□□□□□□□□	driver.window_handles

4.2.2 WebDriver□□

WebDirver□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□　□	□　□	□　□	
back()	□□□□□□□□□□□□□□□□□ □□□□□□□□□□□		driver.back()
close()	□□□□□□□□□		driver.close(

[illegible]	[illegible]	[illegible]	
forward()	[illegible]		driver.forwar
get(url)	[illegible]URL[illegible]	URL[illegible]	driver.get("h
maximize_window()	[illegible]		driver.maxim
quit()	[illegible]driver[illegible]		driver.quit()
refresh()	[illegible]		driver.refresl
switch_to_active_element()	[illegible]		driver.switch
switch_to_alert()	[illegible]		driver.switch
switch_to_default_content()	[illegible]		driver.switch
switch_to-frame(frame_reference)	[illegible] IFRAMES	frame_reference: [illegible]	driver.switch
switch_to_window(window_name)	[illegible]	window_name[illegible]	driver.switch

□ □	□ □	□ □	
implicitly_wait(time_to_wait)	□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□ session□□□□□□□□ execute_async_script □□□□□□□□□ set_script_timeout□□	time_to_wait: □□ □□□□□□□□	
set_page_load_timeout(time_to_wait)	□□□□□□□□□□□□□□□□□ □□	time_to_wait: □□ □□□□□□□□	driver.set_pa
set_script_timeout(time_to_wait)	□□□□□□□□□□□□□□□ execute_async_script □□□□□□	time_to_wait: □□ □□□□□□□□	driver.set_sc

4.3 WebElement□□

□□□□□WebElement□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□div□□

WebElement□□

http://selenium.googlecode.com/git/docs/api/py/webdriver_remote/selenium.webdriver.remote.webelement.html#module-selenium.webdriver.remote.webelement

4.3.1 WebElement□□

□□WebElement□□□□□

□□/□□	□ □	□ □
size	□□□□□□□□	element.size
tag_name	□□□□□HTML□□□□	element.tag_name
text	□□□□□□□□□	element.text

4.3.2 WebElement□□

□□□WebElement□□□□□

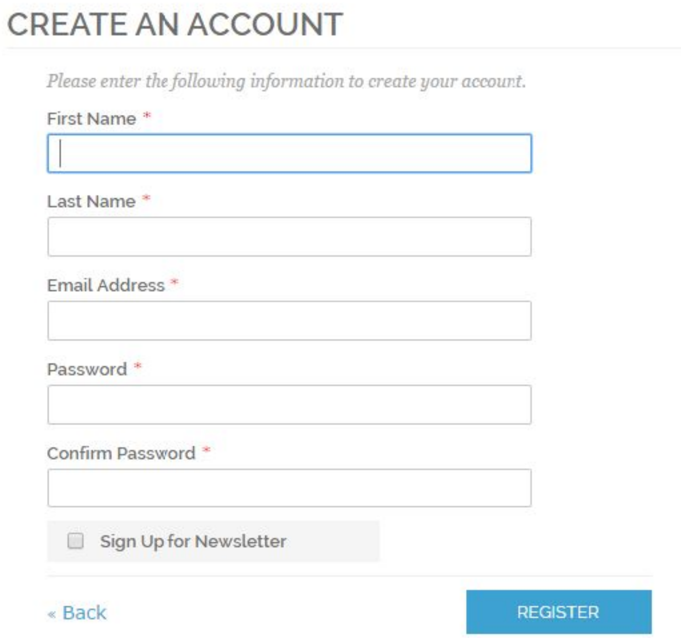
□ □	□ □	□ □	□ □
clear()	□□□□□□□□□□□□□□		element.clear()
click()	□□□□		element.click()
get_attribute(name)	□□□□□□□□	name□□□□□□	element.get_attribute("value") □□ element.get_attribute("maxlength")
is_displayed()	□□□□□□□□□□□□		element.is_displayed()

[illegible]	[illegible]	[illegible]	[illegible]
is_enabled()	[illegible]		element.is_enabled()
is_selected()	[illegible]		element.is_selected()
send_keys(*value)	[illegible]	value[illegible]	element.send_keys("foo")
submit()	[illegible]		element.submit()
value_of_css_property (property_name)	[illegible] CSS[illegible]	property_name[illegible] CSS[illegible]	element.value_of_css_property ("backgroundcolor")

4.4 [illegible]

[illegible]WebElement[illegible]HTML[illegible]

[illegible]WebElement[illegible]WebElement[illegible]



[illegible]5[illegible]

[illegible]1[illegible] RegisterNewUser[illegible]

```
from selenium import webdriver
import unittest

class RegisterNewUser(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Firefox
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

        # navigate to the application home page
        self.driver.get("http://demo.magentocommerce.com/")
```

[illegible]2[illegible] test_register_new_user(self) [illegible]RegisterNewUser[illegible]

[illegible]3[illegible]

```
def test_register_new_user(self):
   driver = self.driver

   # click on Log In link to open Login page
   driver.find_element_by_link_text("Log In").click()
```

4.4.1 [illegible]

[illegible]visible[illegible]TRUE[illegible]is_displayed() [illegible]TRUE[illegible]FALSE[illegible]is_enabled() [illegible]TRUE[illegible]FALSE[illegible]

[illegible]is_displayed()[illegible]is_enabled()[illegible]

```
# get the Create Account button
      create_account_button = driver.find_element_by_xpath("//
button[@title='Create an Account']")

      # check Create Account button is displayed and enabled
      self.assertTrue(create_account_button.is_displayed() and
                   create_account_button.is_enabled())
```

[illegible]WebDriver [illegible] title [illegible]

```
# click on Create Account button. This will display
# new account
create_account_button.click()

# check title
self.assertEquals("Create New Customer Account -
  Magento Commerce Demo Store", driver.title)
```

[illegible] find_element_by_* [illegible]

```
# get all the fields from Create an Account form
first_name = driver.find_element_by_id("firstname")
last_name = driver.find_element_by_id("lastname")
```

```
email_address = driver.find_element_by_id("email_address")
news_letter_subscription =
  driver.find_element_by_id("is_subscribed")
password = driver.find_element_by_id("password")
confirm_password = driver.find_element_by_id("confirmation")
submit_button =
  driver.find_element_by_xpath("//button[@title='Submit']")
```

4.4.2 □□□□□□□□□

get_attribute()□□255□□□□□□□□□maxlength□□□□□□□□□□□□□□□□□□255□

```
<input type="text" id="firstname" name="firstname" value=""
  title="First Name" maxlength="255" class="input-text required-entry">
```

□□□□□□□□□get_attribute()□□□□□maxlength □□□□□□□□

□1□□□□□□□□□□□□□□□□get_attribute()□□□

```
# check maxlength of first name and last name textbox
self.assertEqual("255", first_name.get_attribute("maxlength"))
self.assertEqual("255", last_name.get_attribute("maxlength"))
```

□2□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
# check all fields are enabled
       self.assertTrue(first_name.is_enabled()and last_name.is_enabled()
          and email_address.is_enabled() and
news_letter_subscription.is_enabled()
          and password.is_enabled() and confirm_password.is_enabled()
          and submit_button.is_enabled())
```

4.4.3 is_selected()□□

is_selected() □□

□□□□□□□□□□□□□WebElement □ click() □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Sign UP for Newsletter □□□□□□□□□□□□□□□□□□□□□□□

```
# check Sign Up for Newsletter is unchecked
      self.assertFalse(news_letter_subscription.is_selected())
```

4.4.4 clear()[illegible]send_keys()[illegible]

clear() [illegible] send_keys()[illegible]send_keys() [illegible]

[illegible]1[illegible]send_keys() [illegible]

```
# fill out all the fields
first_name.send_keys("Test")
last_name.send_keys("User1")
news_letter_subscription.click()
email_address.send_keys("TestUser_150214_2200@example.com")
password.send_keys("tester")
confirm_password.send_keys("tester")
```

[illegible]2[illegible]

[illegible]text [illegible]

```
# check new user is registered
self.assertEqual("Hello, Test User1!", driver.find_
   element_by_css_selector("p.hello > strong").text)
self.assertTrue(driver.find_element_by_link_text("LogOut").is_displayed())
```

[illegible]3[illegible] Create An Account [illegible]

```
from selenium import webdriver
from time import gmtime, strftime
import unittest

class RegisterNewUser(unittest.TestCase):
   def setUp(self):
      self.driver = webdriver.Firefox()
      self.driver.implicitly_wait(30)
      self.driver.maximize_window()

      # navigate to the application home page
      self.driver.get("http://demo.magentocommerce.com/")
```

```
   def test_register_new_user(self):
      driver = self.driver

      # click on Log In link to open Login page
      driver.find_element_by_link_text("ACCOUNT").click()
      driver.find_element_by_link_text("My Account").click()

      # get the Create Account button
      create_account_button = \
         driver.find_element_by_link_text("CREATE AN ACCOUNT")

      # check Create Account button is displayed
      # and enabled
      self.assertTrue(create_account_button.
        is_displayed() and
        create_account_button.is_enabled())

      # click on Create Account button. This will
      # display new account
      create_account_button.click()

      # check title
      self.assertEquals("Create New Customer Account", driver.title)

      # get all the fields from Create an Account form
      first_name = driver.find_element_by_id("firstname")
      last_name = driver.find_element_by_id("lastname")
      email_address = driver.find_element_by_id("email_address")
      password = driver.find_element_by_id("password")
      confirm_password = driver.find_element_by_id("confirmation")
      news_letter_subscription =
driver.find_element_by_id("is_subscribed")
      submit_button = driver.\find_element_by_xpath
("//button[@title='Register']")

      # check maxlength of first name and
      # last name textbox
      self.assertEqual("255", first_name.get_attribute("maxlength"))
      self.assertEqual("255", last_name.get_attribute("maxlength"))

      # check all fields are enabled
      self.assertTrue(first_name.is_enabled()
        and last_name.is_enabled()
        and email_address.is_enabled() and
        news_letter_subscription.is_enabled() and
        password.is_enabled() and
        confirm_password.is_enabled()
        and submit_button.is_enabled())

      # check Sign Up for Newsletter is unchecked
      self.assertFalse(news_letter_subscription. is_selected())

      user_name = "user_" + strftime ("%Y%m%d%H%M%S", gmtime())

      # fill out all the fields
      first_name.send_keys("Test")
```

```
        last_name.send_keys(user_name)
        news_letter_subscription.click()
        email_address.send_keys(user_name + "@example.com")
        password.send_keys("tester")
        confirm_password.send_keys("tester")

        # click Submit button to submit the form
        submit_button.click()

        # check new user is registered
        self.assertEqual("Hello, Test " + user_name + "!",
           driver.find_element_by_css_selector("p.hello >strong").text)
        driver.find_element_by_link_text("ACCOUNT").click()
        self.assertTrue(driver.find_element_by_link_text ("Log
Out").is_displayed())

    def tearDown(self):
        self.driver.quit()

if __name__ == "__main__":
    unittest.main(verbosity=2)
```

4.5 □□□□□

Selenium WebDriver□□□□□□Select□□□

□□□□□□□□□□HTML□<select> □□□□□□□□□□□□<select>□□<option>□□□□□□□□HTML□□□

```
<select id="select-language" title="Your Language"
  onchange="window.location.href=this.value">
  <option value="http://demo.magentocommerce.com/?
    ___store=default&amp;___from_store=default"
    selected="selected">English</option>
  <option value="http://demo.magentocommerce.com/?
    ___store=french&amp;___from_store=default">French</option>
  <option value="http://demo.magentocommerce.com/?
```

```
    ___store=german&amp;___from_store=default">German</option>
</select>
```

[illegible]<option> [illegible]<option>[illegible]URL[illegible]French[illegible]

```
<option value="http://demo.magentocommerce.com/customer/
  account/create/?___store=french&amp;
  ___from_store=default">French</option>
```

4.5.1 Select[illegible]

Select [illegible]Selenium[illegible]

[illegible]Select[illegible]

http://selenium.googlecode.com/git/docs/api/py/webdriver_support/selenium.webdriver.support.select.html#module-selenium.webdriver.support.select

4.5.2 Select[illegible]

Select[illegible]

[illegible]/[illegible]	[illegible] [illegible]	[illegible] [illegible]
all_selected_options	[illegible]	select_element.all_selected_options
first_selected_option	[illegible]	select_element.first_selected_option

□□/□□	□ □	□ □
options	□□□□□□□□□□□□□□□	select_element.options

4.5.3 Select□□

Select□□□□□□□□□□

□□/□□	□ □	□ □	□ □
deselect_all()	□□□□□□□□□□□□□□□		select_element.deselect_all()
deselect_by_index(index)	□□□□□□□□□□□□□□□	index□□□□□□□□□□□□	select_element.deselect_by_index(1)
deselect_by_value(value)	□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□	value□□□□□□□□□□□□□□ value□□	select_element.deselect_by_value("foo")

□□/□□	□ □	□ □	□ □
deselect_ by_visible_text(text)	□□□□ □□□□ □□□□ □□□□ □□□□ □□□□	text□□ □□□□□ □□□□□ □□	select_element.deselect_by_visible_text("bar")
select_by_index(index)	□□□□ □□□□ □□□□ □□□□ □	index□ □□□□□ □□□□□ □□	select_element.select_by_index(1)
select_by_value(value)	□□□□ □□□□ □□□□ □□□□ □□□□ □□	value□ □□□□□ □□□□□ value□ □	select_element.select_by_value("foo")
select_by_visible_text(text)	□□□□ □□□□ □□□□ □□□□ □□□□ □□□□	text□□ □□□□□ □□□□□ □□	select_element.select_by_visible_text("bar")

[illegible]

[illegible]8[illegible]options [illegible]

[illegible]

```
def test_language_options(self):
    # list of expected values in Language dropdown
    exp_options = ["ENGLISH", "FRENCH", "GERMAN"]

    # empty list for capturing actual options displayed
    # in the dropdown
    act_options = []

    # get the Your language dropdown as instance of Select class
    select_language = \
        Select(self.driver.find_element_by_id("select-language"))

    # check number of options in dropdown
    self.assertEqual(2, len(select_language.options))

    # get options in a list
    for option in select_language.options:
        act_options.append(option.text)

    # check expected options list with actual options list
    self.assertListEqual(exp_options, act_options)

    # check default selected option is English
    self.assertEqual("ENGLISH", select_language.first_selected_option. text)

    # select an option using select_by_visible text
    select_language.select_by_visible_text("German")

    # check store is now German
    self.assertTrue("store=german" in self.driver.current_url)

    # changing language will refresh the page,
    # we need to get find language dropdown once again
    select_language = \
       Select(self.driver.find_element_by_id("select-language"))
        select_language.select_by_index(0)
```

options[illegible]<option> [illegible]WebElement[illegible]

[illegible]first_selected_option[illegible]/[illegible]

all_selected_options [illegible]

[illegible]**URL**[illegible]

```
# select an option using select_by_visible text
select_language.select_by_visible_text("German")

# check store is now German
self.assertTrue("store=german" in self.driver.current_url)
```

[illegible] select[illegible]select_ [illegible] select_by_visible_text() [illegible]deselect_ [illegible]

4.6 [illegible]

[illegible]JavaScript [illegible]Selenium[illegible]

4.6.1 Alert [illegible]

Selenium WebDriver [illegible]Alert [illegible] JavaScript [illegible]Alert [illegible]

4.6.2 Alert[illegible]

Alert [illegible]

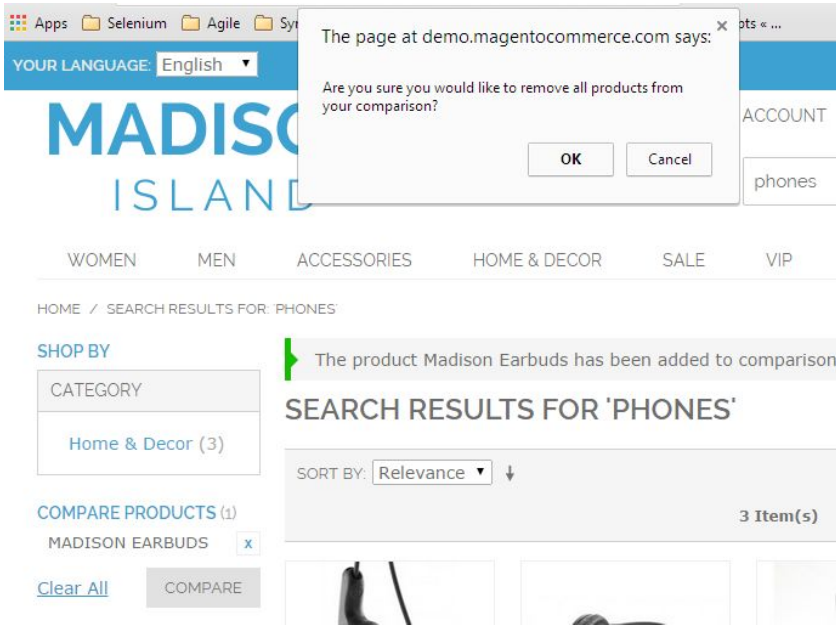
[illegible]/[illegible]	[illegible]	[illegible]
text	[illegible]	alert.text

4.6.3 Alert[illegible]

Alert[illegible]

[illegible]	[illegible]	[illegible]	[illegible]
accept()	[illegible]JavaScript [illegible] OK [illegible]		alert.accept()
dismiss()	[illegible]JavaScript [illegible]		alert.dismiss()
send_ keys(*value)	[illegible]	value[illegible]	alert.send_keys("foo")

[illegible]Alert [illegible]



□□□□□□□□□□□□□□COMPARE PRODUCTS□□□□□□□□□□Clear All□□□□□□□□□□□□□□□□

□□□□□□□□□CompareProducts□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
from selenium import webdriver
import unittest

class CompareProducts(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Firefox()
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()
        self.driver.get("http://demo.magentocommerce.com/")

    def test_compare_products_removal_alert(self):
        # get the search textbox
        search_field = self.driver.find_element_by_name("q")
        search_field.clear()

        # enter search keyword and submit
        search_field.send_keys("phones")
         search_field.submit()

        # click the Add to compare link
        self.driver.\
            find_element_by_link_text("Add to Compare").click()
```

[illegible]Add to Compare [illegible]COMPARE PRODUCTS [illegible]COMPARE PRODUCTS [illegible]Clear All [illegible]“[illegible]”[illegible]Alert[illegible]WebDirver [illegible]Switch_to_alert() [illegible]Alert[illegible]Alert [illegible]OK[illegible]Cancel [illegible]accept() [illegible]

```
    # click on Remove this item link, this will display
    # an alert to the user
    self.driver.find_element_by_link_text("Clear All").click()

    # switch to the alert
    alert = self.driver.switch_to_alert()

    # get the text from alert
    alert_text = alert.text

    # check alert text
    self.assertEqual("Are you sure you would like to
      remove all products from your comparison?", alert_text)

    # click on Ok button
    alert.accept()

    def tearDown(self):
        self.driver.quit()

if __name__ == "__main__":
     unittest.main()
```

4.6.4 [illegible]

[illegible]Selenium WebDriver API [illegible]WebDriver [illegible]

[illegible]	[illegible]	[illegible]	[illegible]

□ □	□ □	□ □	□ □
back()	□□□□□□□□□□□□□□□□□□□□□□□□□□□□	□	driver.back()
forward()	□□□□□□□□□□□□□□□□□□□□□□□□□□□□□	□	driver.forward()
refresh()	□□□□□□□□□□□□	□	driver.refresh()

□□□□□□□□□□□□API□□□□□□□□□□□□□□□□□□□□

```
import unittest
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions

class NavigationTest(unittest.TestCase):
   def setUp(self):
     # create a new Firefox session
     self.driver = webdriver.Chrome()
     self.driver.implicitly_wait(30)
     self.driver.maximize_window()

     # navigate to the application home page
     self.driver.get("http://www.google.com")

   def testBrowserNavigation(self):
     driver = self.driver
     # get the search textbox
     search_field = driver.find_element_by_name("q")
     search_field.clear()

     # enter search keyword and submit
     search_field.send_keys("selenium webdriver")
     search_field.submit()

     se_wd_link = driver.find_element_by_link_text("Selenium WebDriver")
     se_wd_link.click()
     self.assertEqual("Selenium WebDriver", driver.title)

     driver.back()
     self.assertTrue(WebDriverWait(self.driver, 10)
         .until(expected_conditions.title_is
           ("selenium webdriver - Google Search")))

     driver.forward()
     self.assertTrue(WebDriverWait(self.driver, 10)
```

```
        .until(expected_conditions.title_is
          ("Selenium WebDriver")))

      driver.refresh()
      self.assertTrue(WebDriverWait(self.driver, 10)
        .until(expected_conditions.title_is
          ("Selenium WebDriver")))

    def tearDown(self):
      # close the browser window
      self.driver.quit()

if __name__ == '__main__':
    unittest.main()
```

4.7 [illegible]

[illegible]Selenium WebDriver API [illegible]Selenium WebDriver API [illegible]

[illegible]

[illegible]Selenium API[illegible]

[illegible]5[illegible]

[illegible]UI[illegible]
WebDriver[illegible]

[illegible]

- [illegible]
- [illegible]
- [illegible]
- [illegible]

5.1 [illegible]

[illegible]WebDriver[illegible]Ajax[illegible]

[illegible]WebDriver[illegible]DOM[illegible]0[illegible]

[illegible]WebDriver[illegible]WebDriver[illegible]0[illegible]

WebDriver□□□□implicitly_wait()□□□□□□□□□□□□□□□□2□□□□□□ SearchProductTest□□□□□□□unittest□□□□□□□□□□□□□□□□□□□□□setUp() □□□□□□□□□□□□□□□□10□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□WebDirver□ □□□□□□□□□□□□□□□□□10□□□□□10□□□□□□□□□□□□□□□ NoSuchElementException □□□□

```
import unittest
from selenium import webdriver

class SearchProductTest(unittest.TestCase):
   def setUp(self):
      # create a new Firefox session
      self.driver = webdriver.Firefox()
      self.driver.implicitly_wait(30)
      self.driver.maximize_window()

      # navigate to the application home page
      self.driver.get("http://demo.magentocommerce.com/")

   def test_search_by_category(self):
      # get the search textbox
      self.search_field = self.driver.find_element_by_name("q")
      self.search_field.clear()

      # enter search keyword and submit
      self.search_field.send_keys("phones")
      self.search_field.submit()

      # get all the anchor elements which have product names
      # displayed currently on result page using
      # find_elements_by_xpath method
      products = self.driver\
          .find_elements_by_xpath
             ("//h2[@class='product-name']/a")

      # check count of products shown in results
      self.assertEqual(2, len(products))

   def tearDown(self):
      # close the browser window
      self.driver.quit()

if __name__ == '__main__':
   unittest.main(verbosity=2)
```

[illegible]

5.2 [illegible]

[illegible]WebDriver[illegible]

[illegible]WebDriver[illegible]WebDriverWait[illegible]expected_conditions[illegible]

expected_conditions[illegible]

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
import unittest

class ExplicitWaitTests(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Firefox()
        self.driver.get("http://demo.magentocommerce.com/")

    def test_account_link(self):
        WebDriverWait(self.driver, 10)\
            .until(lambda s: s.find_element_by_id("select-
              language").get_attribute("length") == "3")
        account = WebDriverWait(self.driver, 10)\
            .until(expected_conditions.
              visibility_of_element_located
                  ((By.LINK_TEXT, "ACCOUNT")))
        account.click()

    def tearDown(self):
        self.driver.quit()
```

```
if __name__ == "__main__":
    unittest.main(verbosity=2)
```

□□□□□□□□□□□□□□□□□Log In□□□DOM□□□□

□□visibility_of_element_located□□□10□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□TimeoutException□□□

5.3 expected_conditions□

□□□expected_conditions□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□ □ □ □	□ □	□ □	
element_to_be_clickable(locator)	□□□	locator: □□ □by,locator□	WebDriverWait(se 10).until(expectec
element_to_be_selected(element)	□□□□□□□□□□□□	element□ □□ WebElement	subscription = sel WebDriverWait(se element_to_be_se

□□□□	□ □	□ □	
invisibility_of_element_located(locator)	□□□□□□□□ DOM□□□□□□□ □□	locator: □□ □by,locator□	WebDriverWait(se invisibility_of_eler
presence_of_all_elements_located(locator)	□□□□□□□□□□□□ □□□□□□□□□□□□ □□□□□□□□□□ □□□□□□□□□□□□ □ WebElement	locator: □□ □by,locator□	WebDriverWait(se until(expected_co _NAME,"input-text
presence_of_element_located(locator)	□□□□□□□□□□□□ □□□□□□□□□□□□ □□□□□□□□ DOM□□□□ □□ □□□□□□□□□□□□□ □□	locator: □□ □by,locator□	WebDriverWait(se until(expected_co
text_to_be_present_in_element(locator,text_)	□□□□□□□□□□□□ □□□□□□□□□□□□□ □□	locator: □□ □by,locator□ text:□□□□□□ □□□□	WebDriverWait(se until(expected_co language"),"Englis
title_contains(title)	□□□□□□□□□□□□ □□□□□□□□□□□ □□ □□□□□□□□□□□□□ □True□□□□□ False	title: □□□□□□ □□□□□□□□□	WebDriverWait(se NewCustomer Acc

□ □ □ □	□ □	□ □	
title_is(title)	□□□□□□□□□□ □□□□□□ □□□□□□□□□□ □True□□□□□ False	title:□□□□□	WebDriverWait(se CustomerAccount
visibility_of(element)	□□□□□□□□□ DOM□□□□□ □□□□□□□□□□ 0□ □□□□□□□□□□ □□□□□□□□□□ WebElement	element □ □□ WebElement	first_name = self.d 10). until(expecte
visibility_of_element_located(locator)	□□□□□□□□□□ □□□□□□□□□ DOM□□□□□ □□□□□□□□□□ 0□ □□□□□□□□□□ □□□□□ WebElement	locator: □□ □by,locator□	WebDriverWait(se until(expected_co

□□□□□□□□□□□□□□□□□□□□□□□□ http://selenium.googlecode.com/git/docs/api/py/webdriver_support/selenium.webdriver. support.expected_conditions.html#module-selenium. webdriver.support. expected_conditions□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

5.3.1 [illegible]

[illegible]expected_conditons[illegible]Ajax [illegible]Log In[illegible]Create an Account[illegible]Create an Account[illegible]

```
def test_create_new_customer(self):
    # click on Log In link to open Login page
    self.driver.find_element_by_link_text("ACCOUNT").click()

    # wait for My Account link in Menu
    my_account = WebDriverWait(self.driver, 10)\
        .until(expected_conditions.visibility_of_element_located((By.
    LINK_TEXT, "My Account")))
    my_account.click()

    # get the Create Account button
    create_account_button = WebDriverWait(self.driver, 10)\
        .until(expected_conditions.element_to_be_clickable((By.LINK_
        TEXT, "CREATE AN ACCOUNT")))

    # click on Create Account button. This will displayed new account
    create_account_button.click()
    WebDriverWait(self.driver, 10)\
        .until(expected_conditions.title_contains("Create New Customer
Account"))
```

[illegible]element_to_be_clickable[illegible]

[illegible]title_contains[illegible]

5.3.2 [illegible]Alerts

[illegible]JavaScript[illegible]alert_is_present[illegible]

```
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions
import unittest

class CompareProducts(unittest.TestCase):
   def setUp(self):
      self.driver = webdriver.Firefox()
      self.driver.get("http://demo.magentocommerce.com/")

   def test_compare_products_removal_alert(self):
      # get the search textbox
      search_field = self.driver.find_element_by_name("q")
      search_field.clear()

      # enter search keyword and submit
      search_field.send_keys("phones")
      search_field.submit()

      # click the Add to compare link
      self.driver.\
         find_element_by_link_text("Add to Compare").click()

      # wait for Clear All link to be visible
      clear_all_link = WebDriverWait(self.driver, 10)\
         .until(expected_conditions.visibility_of_element_
         located((By.LINK_TEXT, "Clear All")))

      # click on Clear All link,
      # this will display an alert to the user
      clear_all_link.click()

      # wait for the alert to present
      alert = WebDriverWait(self.driver, 10)\
         .until(expected_conditions.alert_is_present())

      # get the text from alert
      alert_text = alert.text

      # check alert text
      self.assertEqual("Are you sure you would like
  to remove all products from your comparison?", alert_text)
      # click on Ok button
      alert.accept()

   def tearDown(self):
      self.driver.quit()

if __name__ == "__main__":
   unittest.main(verbosity=2)
```

[illegible]Alert_is_present [illegible]10[illegible]

5.4 [illegible]

[illegible]expected_conditions [illegible]WebDriverWait [illegible]

[illegible]

```
def testLoginLink(self):
    WebDriverWait(self.driver, 10).until
      (lambda s: s.find_element_by_id
      ("select-language").get_attribute("length") == "3")

    login_link = WebDriverWait
      (self.driver, 10).until(expected_conditions.
       visibility_of_element_located((By.LINK_TEXT,"Log In")))
       login_link.click();
```

[illegible]Python[illegible]lambda[illegible]WebDriverWait[illegible]10[illegible]Select Language[illegible]8[illegible]Ajax[illegible]

5.5 [illegible]

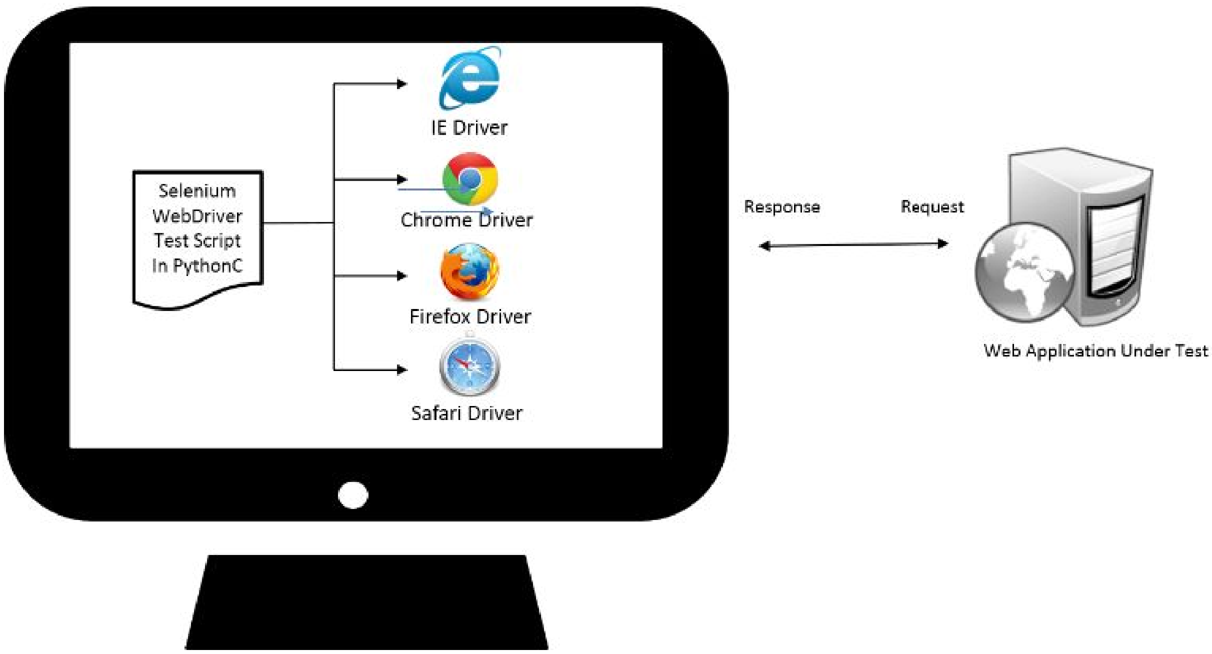
[illegible]expected_conditions[illegible]

WebDriverWait□□□□□□□□□□□□□□□□□□□□expected_conditions□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□RemoteWebDriver□Selenium Server□□□□□□□□□□□□□□□Selenium Grid□□□□□□□□□□□□□□□□□□□□

[illegible]6[illegible] [illegible]

Selenium[illegible]Web[illegible]Selenium WebDriver[illegible]



[illegible]

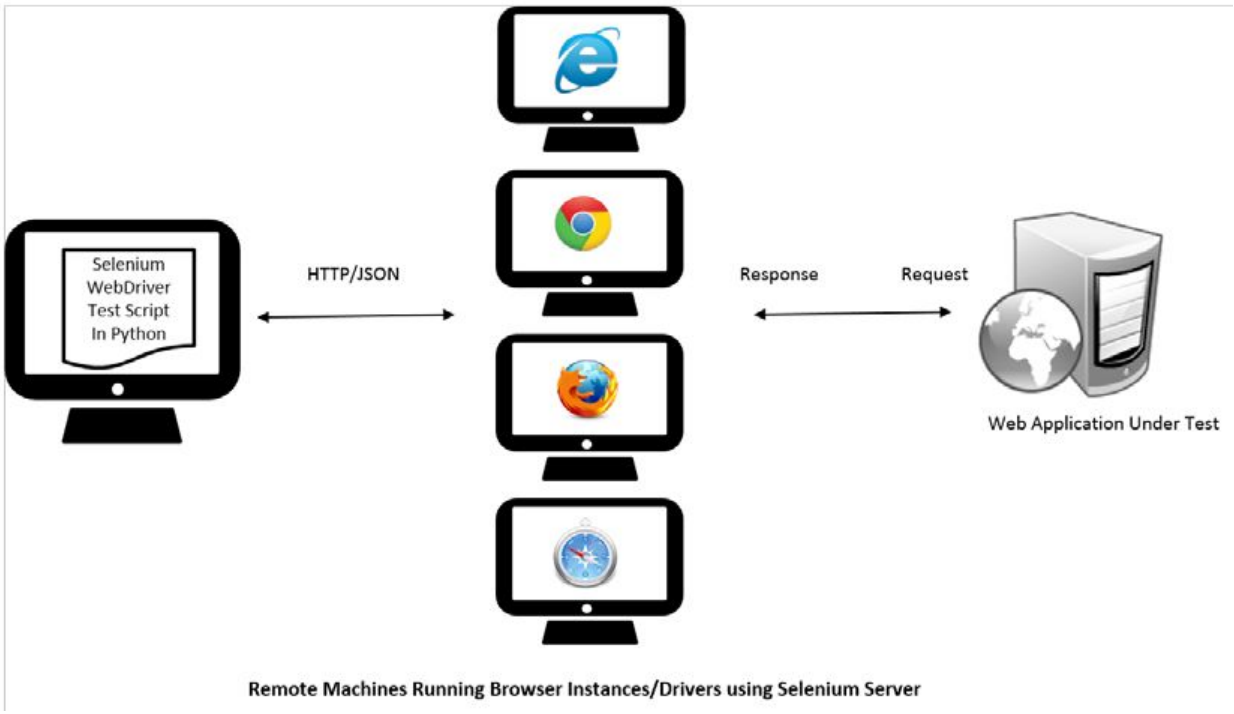
[illegible]

- Selenium Standalone Server [illegible]
- [illegible]Remote [illegible]Selenium Standalone Server [illegible]
- [illegible]Selenium Standalone Server[illegible]

- □Selenium Standalone Server□□□□□□□□□
□□□□□□□□Grid□
- □□□□□□□□□□□□□□□□Grid□□□□□□
- □□Sauce Labs □BrowserStack□□□□□□□□

6.1 Selenium Standalone Server

Selenium Standalone Server □□ Selenium□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□RemoteWebDriver□□□□□Selenium Standalone Server□□□□□□□□□□□□□□□□RemoteWebDriver□□□□□□□□□□Selenium□□□□□□□□□□□□□□RemoteWebDriver□□□□□□□□□Selenium Server□□□Appium□□□□□□□□□□□□□□□Selenium Server□□□□□□□□□□□□□□□□□□□□□□□□□□



6.1.1 □□Selenium Standalone Server

Selenium Standalone Server□□JAR□□□□□□□□□□http://docs.seleniumhq.org/download/□□□Selenium Server□□□□Selenium RC Server□□□□□□□□□□□□Selenium Server□□□□□□2.41.0□□□□□□□□Selenium Standalone Server□JAR□□□□□□□□□□□□□□□□□□□

Selenium Standalone Server□□Java □□□□□□□□□□□□□□□□□□□□□□□□□□□□JRE(Java Runtime Environment)□□□□Selenium Standalone Server□□□□□□□□□□□□□□□□ JRE6□□□□□□□□

6.1.2 □□Selenium Standalone Server

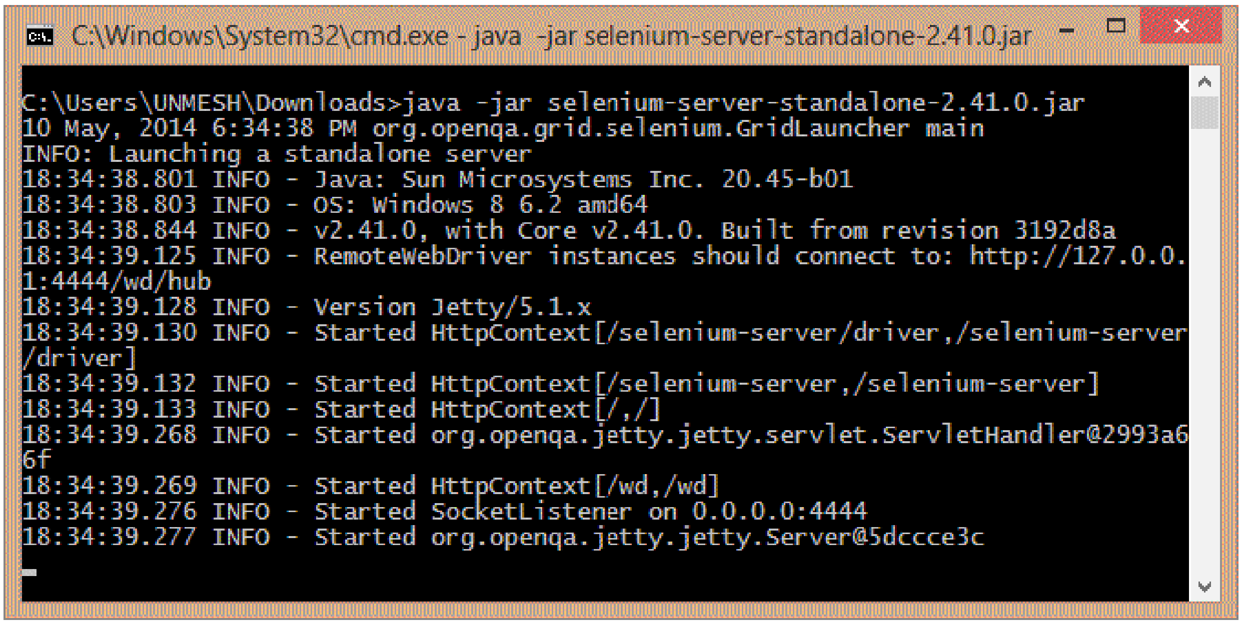
Selenium Standalone Server□□□□□□□□ □□□□□□□□□□□□□□□Standalone□□□□□□□□□□ □□□□□□Selenium Standalone Server□JAR□□ □□□□□□□□□□□□□□□□□□□□□Selenium Server□□□ □Windows 8□□□Selenium Standalone Server□□□□

```
java –jar selenium-server-standalone-2.41.0.jar
```

Selenium Server 4444)

http://<remote-machine-ip>:444

Selenium Server



Selenium Server HTTP Server

http://<remote-machine-ip>:4444/wd/hub/static/resource/hub.html

□□□□□□□□□□□Selenium Server□□□□□□□□□□□□□□□

6.2 □Selenium Standalone Server□□□□□□

□□Selenium Server□□□□□□□□□□□□□RemoteWebDriver□□□Selenium Python□□Remote□□□□□□□□□□Selenium Server□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Selenium Server□□□

□□Remote□□□□□□□□□□□desired_capabilities□□□□□□□□□□□□□□□□□□□□□

□Selenium Standalone Server□□□desired_capabilities□□□□□

```
desired_caps = {}
desired_caps['platform'] = 'WINDOWS'
desired_caps['browserName'] = 'firefox
```

□□□□□□□□□Remote□□□□□□□desired_capabilities□□□□□□□□□□□□□□□□Selenium Server□□Windows□□□□Firefox□□□□□□□□

```
self.driver =
webdriver.Remote('http://192.168.1.103:4444/wd/hub
', desired_caps)
```

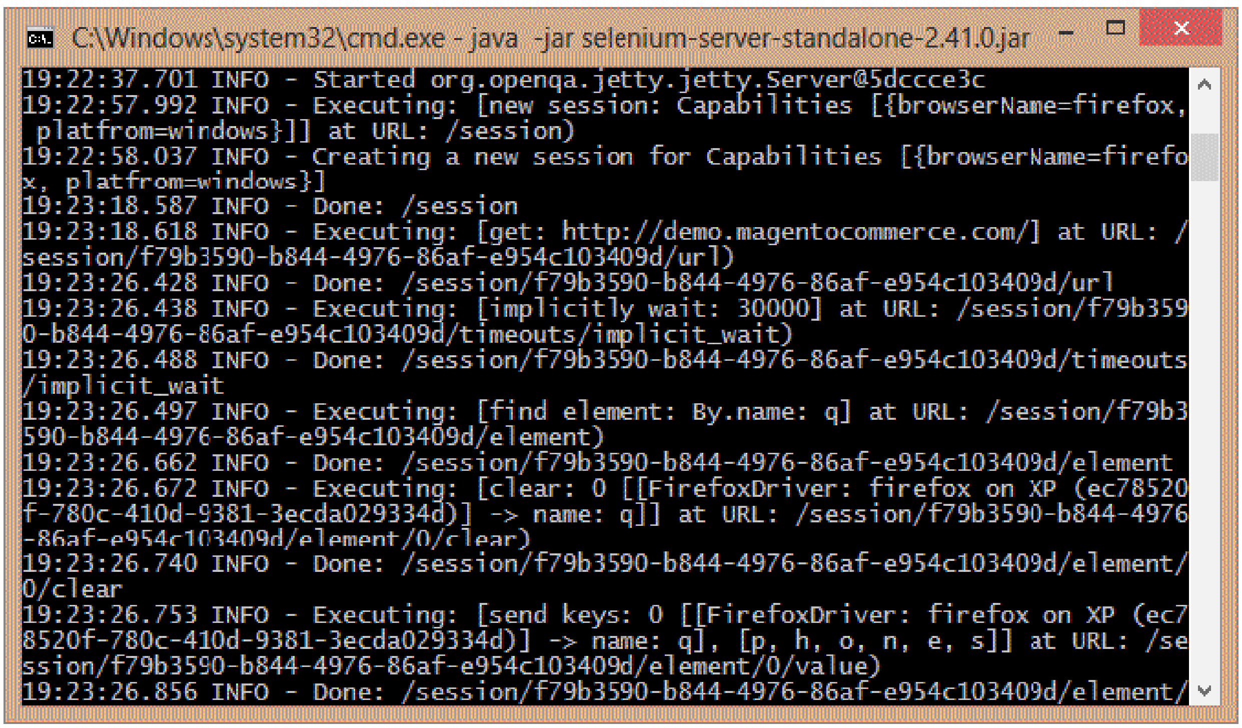
□□□□□Remote□□□Firefox driver□□□□□□□□□□□□□□

```
import unittest
from selenium import webdriver

class SearchProducts(unittest.TestCase):
    def setUp(self):
```

```
        desired_caps = {}
        desired_caps['platform'] = 'WINDOWS'
        desired_caps['browserName'] = 'firefox'

        self.driver = \

webdriver.Remote('http://192.168.1.102:4444/wd/hub
', desired_caps)

self.driver.get('http://demo.magentocommerce.com/'
)
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

    def testSearchByCategory(self):
        # get the search textbox
        self.search_field =
self.driver.find_element_by_name('q')
        self.search_field.clear()

        # enter search keyword and submit
        self.search_field.send_keys('phones')
        self.search_field.submit()

        # get all the anchor elements which have
product names
        # displayed currently on result page using
        # find_elements_by_xpath method
        products = self.driver\
            .find_elements_by_xpath('//h2[@class=\'p
roduct-name\']/a')

        # check count of products shown in results
        self.assertEqual(2, len(products))

    def tearDown(self):
        # close the browser window
```

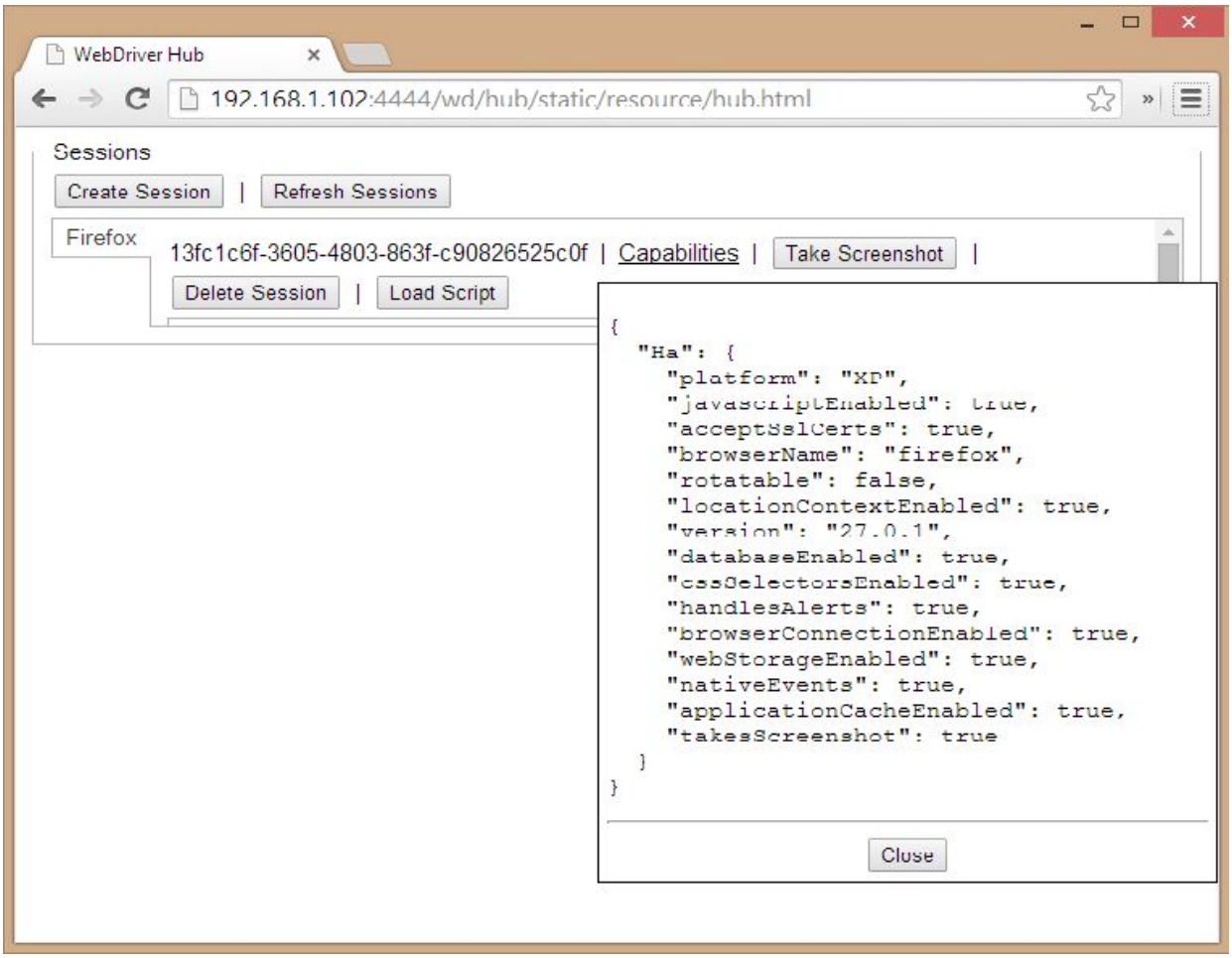
```
        self.driver.quit()

if __name__ == '__main__':
    unittest.main()
```

□□□□□□□□□□□□□□□□□□Selenium Server□□□□□□□□□□□□□□□□□□□□□□□□Selenium Server□□□



□□□□□□http:// <remote-machine-ip>:4444/wd/hub/static/resource/hub.html

□□□□□□□□□□□□□□□□□□□□□□□□□□□Capabilities□□□
□□□□□□□□□□□□Capabilities□□□□□□□□□□□□



6.2.1 □□IE□□

Selenium Server□Firefox□□□□□□□□Firefox□□□□□□Internet Explorer□IE□□□□□□□□□□□□Selenium Server□□□IE driver□□□□□□□

□□□□□□□□□□□□□□□□webdriver.ie.driver□□□□□IE driver□□□□□□□□□□□

```
java -
Dwebdriver.ie.driver="C:\SeDrivers\IEDriverServer.
exe" –jar
selenium-server-standalone-2.41.0.jar
```

□□□□IE driver□□□□Selenium Server□□□□□□□□□□□IE□□□□□□□□□□

6.2.2 □□Chrome□□

□IE driver□□□□□□Chrome□□□□□□□□□□□□ Chrome driver□□□□□□□□□□□□□ webdriver.chrome.driver□□□□Chrome driver□□□□□□□□□□□

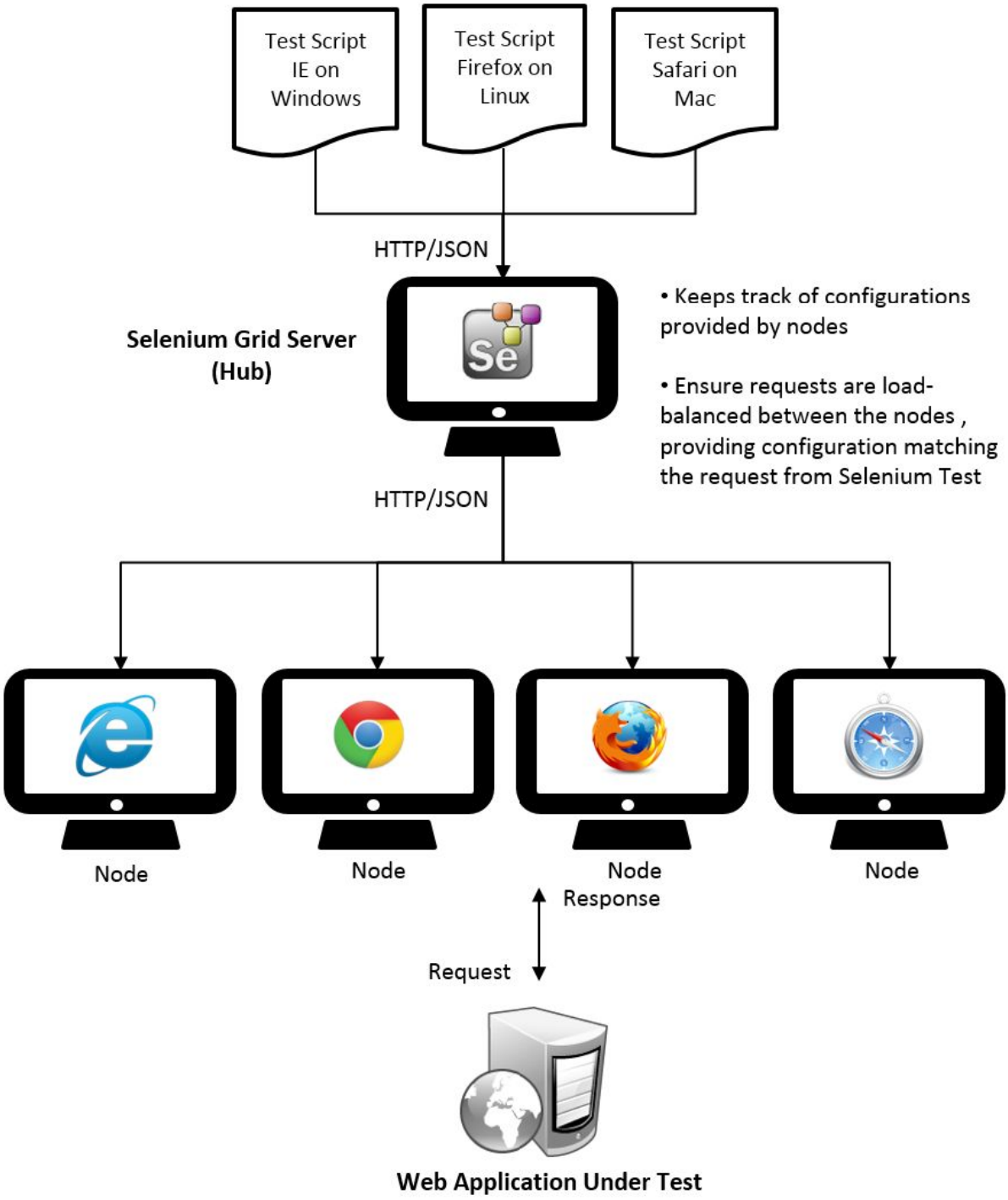
```
java -
Dwebdriver.ie.driver="C:\SeDrivers\IEDriverServer.
exe" -Dwebdriver.
chrome.driver="C:\SeDrivers\chromedriver.exe" -jar
selenium-server-standalone-2.41.0.jar
```

[illegible]Selenium Server[illegible]Internet Explorer[illegible]Chrome[illegible]

6.3 Selenium Grid

Selenium Grid[illegible]Grid[illegible]

Selenium Grid[illegible]Grid[illegible]


Test Script
IE on
Windows
Test Script
Firefox on
Linux
Test Script
Safari on
Mac
HTTP/JSON
Selenium Grid Server
(Hub)
• Keeps track of configurations provided by nodes
• Ensure requests are load-balanced between the nodes , providing configuration matching the request from Selenium Test
HTTP/JSON
Node
Node
Node
Node
Response
Request
Web Application Under Test

6.3.1 □□hub

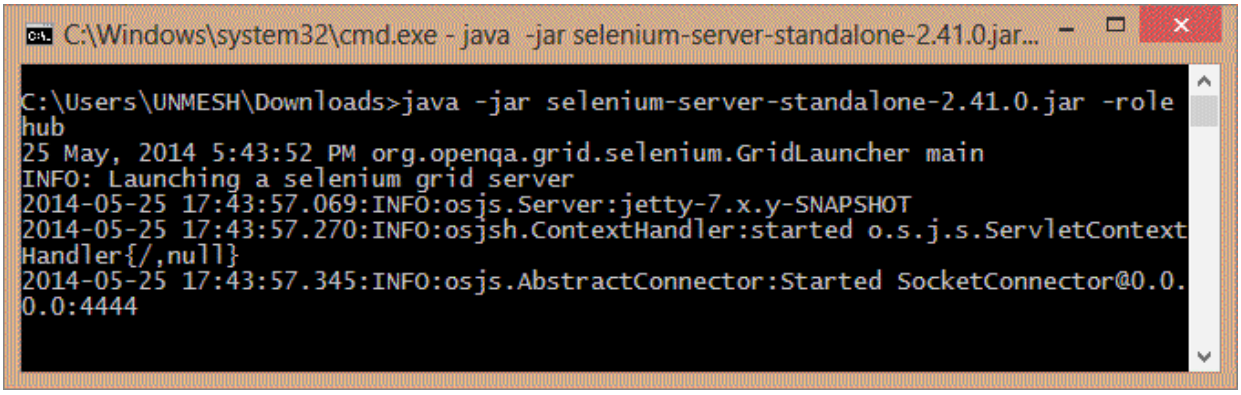
□□□□□□□□□□Selenium Server□□□□hub□
hub□□□□□□□□□□□□□□□□□□□□□□□□□□□□slave□□□□
□□□□□□□□□□□□hub□□□□□□□JSON Wire
Protocol□□□□Remote□□hub□□□□□Selenium□
□□ □□□□JSON Wire Protocol□□□□□□□
https://code.google.com/p/selenium/wiki/Js
onWireProtocol□

hub□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□Selenium Server□□□Grid□□□□
□□□□□□□□□□□□□□□□□□□□□

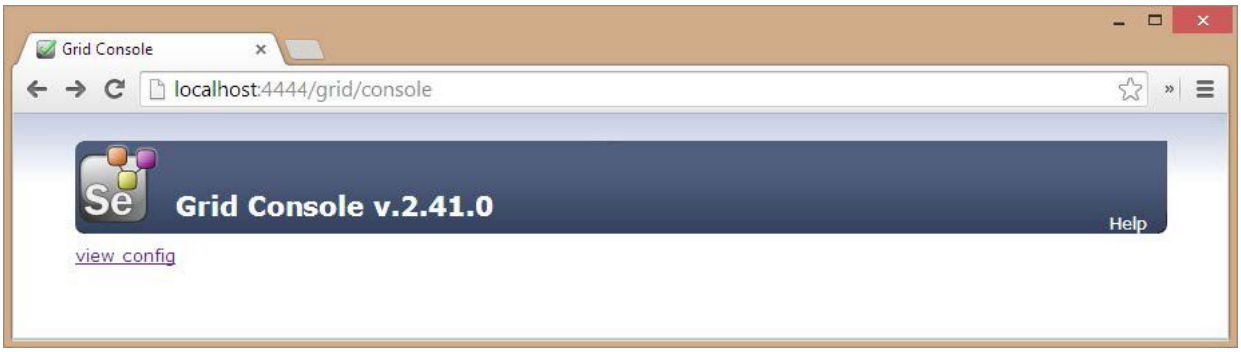
□□□□□□□□□□□□□□□□□□□□□□□□□□□Selenium
Standalone Server□□hub□□□□Grid Server□□
□□□□□□□□□□□□□□/□□□□□□□□□□□Selenium
Server JAR□□□□□□□□□□□□□□hub□□□□Server□

```
java -jar selenium-server-standalone-2.25.0.jar -
port 4444 -role hub
```

□□□□□Server□□□hub□□Grid Server□□□□
□□□□□□□□□-*role*□□□□□“hub”□□□□□□□□□
Windows□□□□□Server□□□□□□□□□□□□□□□□□□□□



□□□□hub□□□□Server□□□□□□□□Grid Server□□□□□□□□□□□□□□Grid□□□□□□□□□□□□□



6.3.2 □□□□

□□□□□□Selenium Server□□Grid Server□□□□□□□□Server□□□□□□□□□

6.3.2.1 □□IE□□

□□□□□Internet Explorer□Windows□□□□□□□□□□□□□□□□□□□□□□□□Selenium Server JAR□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Grid□

```
java -
Dwebdriver.ie.driver="C:\SeDrivers\IEDriverServer.
exe" -jar
selenium-server-standalone-2.41.0.jar -role
webdriver -browser
"browserName=internet
explorer,version=10,maxinstance=1,platform=WINDOWS
"
-hubHost 192.168.1.103 –port 5555
```

□□□□□□□Grid□□□□□□□-role□□□□□“webdriver”□□□□□-browser□□□□□□□□□□□□□□□□□□□browserName□□□□□Internet Explorer□□□10□maxinstance□1□□□□

Windows□□□maxinstance□□□□Grid□□□□□□□□□□□□□□□

□□□□□□□□hub□Grid Server□□□□□□□□□-hubHost□□□Grid Server□□□□□IP□□□ □□□□□□□□□□□□□□□hub□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□Grid□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□JSON□□□□□□□□□□□□□□□□JSON□□□□□□□□□□

```
{
  "class":
"org.openqa.grid.common.RegistrationRequest",
  "capabilities": [
   {
   "seleniumProtocol": "WebDriver",
   "browserName": "internet explorer",
   "version": "10",
   "maxInstances": 1,
   "platform" : "WINDOWS"
   }
  ],
  "configuration": {
   "port": 5555,
   "register": true,
   "host": "192.168.1.103",
   "proxy": "org.openqa.grid.selenium.proxy.
DefaultRemoteProxy",
   "maxSession": 2,
   "hubHost": "192.168.1.100",
   "role": "webdriver",
   "registerCycle": 5000,
   "hub":
"http://192.168.1.100:4444/grid/register",
   "hubPort": 4444,
   "remoteHost": "http://192.168.1.102:5555"
  }
}
```

□□□□□□□□□□□□□□JSON□□□□“selenium-node-win-ie10.cfg.json”□□□□□□□JSON□□□□□□□□Server□□□□

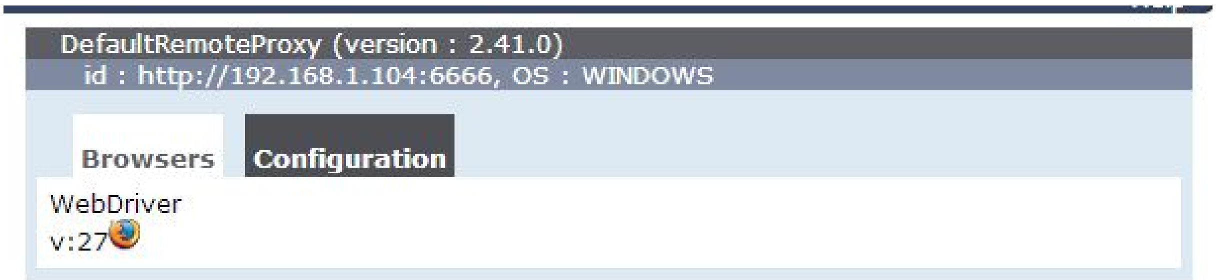
```
java -
Dwebdriver.ie.driver="C:\SeDrivers\IEDriverServer.
exe"-jar
selenium-server-standalone-2.41.0.jar -role
webdriver -nodeConfig
selenium-node-win-ie10.cfg.json
```

6.3.2.2 □□Firefox□□

□□□□□□Firefox□□□□□□□□□□□□□□□□□□□Selenium Server JAR□□□□□□□□□□□□□□□□□□□□□□□Grid□

```
java -jar selenium-server-standalone-2.41.0.jar -
role webdriver -browser
"browserName=firefox,version=27,maxinstance=2,plat
form=WINDOWS" –hubHost
localhost –port 6666
```

□□□□□□□maxinstance□□2□□□□□□Grid□□□□□□□Firefox□□□Firefox□□□□□□Grid□□□□□□□□□□□□□□□□

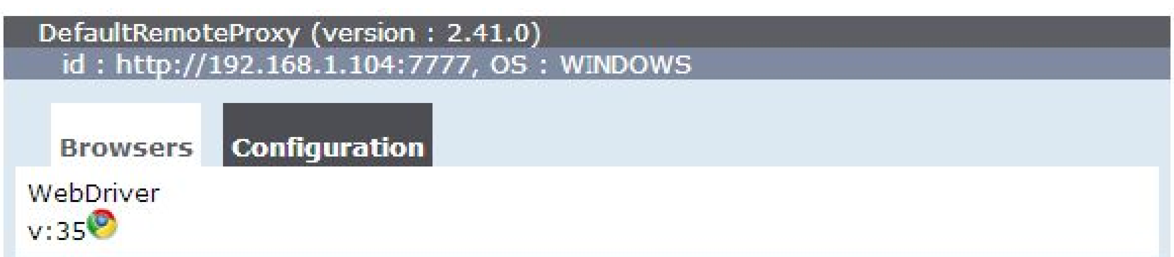


6.3.2.3 □□Chrome□□

□□□□□Chrome□□□□□□□□□□□□□□□□□□□Selenium Server JAR□□□□□□□□□□□□□□□□□□□□□□□Grid□

```
java -
Dwebdriver.chrome.driver="C:\SeDrivers\chromedrive
r.exe" -jar
selenium-server-standalone-2.41.0.jar -role
webdriver -browser
"browserName=chrome,version=35,maxinstance=2,platf
orm=WINDOWS" -hubHost localhost -port 7777
```

□□□Chrome□□□□□Grid□□□□□□□□□□□

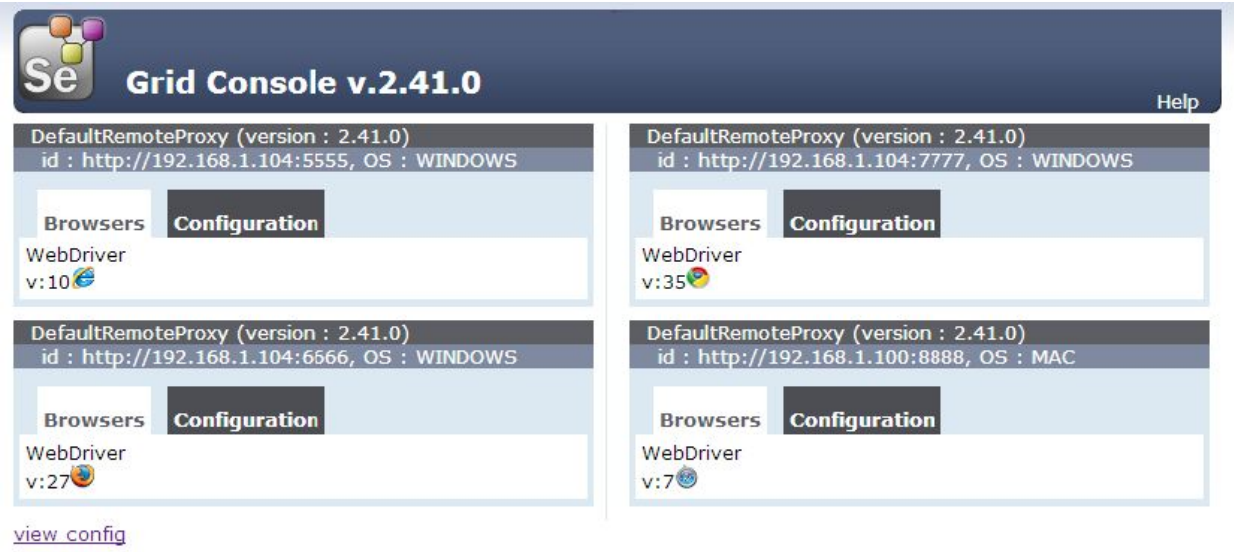


6.4 Mac OS X□Safari□□

□□□□□Windows□□□□□IE□Firefox□Chrome□□□□□□□□□Mac□□□□□Safari□□□□□□□□□□□□□□□Selenium Server JAR□□□□□□□□□□□□□□□□□□□□□□□□□□□Grid□

```
java -jar selenium-server-standalone-2.41.0.jar -role webdriver -browser "browserName=safari,version=7,maxinstance=1,platform=MAC" -hubHost 192.168.1.104 -port 8888
```

□□□□□□□□□Grid□□□□□□□□□□□□□□□



[illegible]Selenium Grid[illegible]
Grid[illegible]

6.5 [illegible]Grid[illegible]

[illegible]Grid[illegible]desired_capabilities[illegible]desired_capabilities[illegible]

[illegible]1[illegible]Windows[illegible]Chrome[illegible]

```
python grid_test.py WINDOWS chrome
```

[illegible]2[illegible]Mac[illegible]Safari[illegible]

```
python grid_test.py MAC safari
```

[illegible]3[illegible]PLATFORM[illegible]BROWSER[illegible]

```
class SearchProducts(unittest.TestCase):
    PLATFORM = 'WINDOWS'
    BROWSER = 'firefox'
```

[illegible]4[illegible]setUp()[illegible]desired_capabilities[illegible]

```
desired_caps = {}
desired_caps['platform'] = self.PLATFORM
desired_caps['browserName'] = self.BROWSER
```

[illegible]5[illegible]PLATFORM[illegible]BROWSER[illegible]

```
if __name__ == '__main__':
    if len(sys.argv) > 1:
        SearchProducts.BROWSER = sys.argv.pop()
        SearchProducts.PLATFORM = sys.argv.pop()
    unittest.main()
```

□6□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□

```
import sys
import unittest
from selenium import webdriver

class SearchProducts(unittest.TestCase):
    PLATFORM = 'WINDOWS'
    BROWSER = 'firefox'

    def setUp(self):
        desired_caps = {}
        desired_caps['platform'] = self.PLATFORM
        desired_caps['browserName'] = self.BROWSER

        self.driver = \

webdriver.Remote('http://192.168.1.104:4444/wd/hub
', desired_caps)

self.driver.get('http://demo.magentocommerce.com/'
)
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

    def testSearchByCategory(self):
        # get the search textbox
        self.search_field =
self.driver.find_element_by_name('q')
        self.search_field.clear()

        # enter search keyword and submit
        self.search_field.send_keys('phones')
        self.search_field.submit()
```

```
        # get all the anchor elements which have
product names
        # displayed currently on result page using
        # find_elements_by_xpath method
        products = self.driver.\

find_elements_by_xpath('//h2[@class=\'product-
name\']/a')

        # check count of products shown in results
        self.assertEqual(2, len(products))

    def tearDown(self):
        # close the browser window
        self.driver.quit()

if __name__ == '__main__':
    if len(sys.argv) > 1:
       SearchProducts.BROWSER = sys.argv.pop()
       SearchProducts.PLATFORM = sys.argv.pop()
    unittest.main(verbosity=2)
```

□7□□□Grid□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
python grid_test.py MAC safari
```

6.6 □□□□□□

[illegible]Grid[illegible][illegible]Grid[illegible]

[illegible]lab[illegible]lab[illegible]Sauce Labs[illegible]BrowserStack[illegible]400[illegible]Selenium WebDriver[illegible]

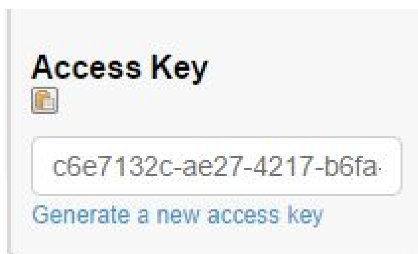
[illegible]Sauce Labs[illegible]BrowserStack[illegible]

[illegible]Sauce Labs[illegible]

[illegible]1[illegible]Sauce Labs[illegible]Sauce Labs[illegible]https://saucelabs.com/[illegible]

□□□□□□□□□Sauce Labs□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□

□2□□□□□Sauce Labs□□□□□□□□□□□□□□□



□3□□□□□Grid□□□□□□□□□□□□□□□□□□□□□□
□Sauce Labs□□

□4□□□□□□□□□Sauce□□□□□□□□□□□Grid□□
□Sauce Grid□□□□□□□□□□□□□□□□□□□□□□□

```
import sys
import unittest
from selenium import webdriver

class SearchProducts(unittest.TestCase):
    PLATFORM = 'WINDOWS'
    BROWSER = 'phantomjs'
    SAUCE_USERNAME = 'upgundecha'
    SUACE_KEY = 'c6e7132c-ae27-4217-b6fa-
3cf7df0a7281'
```

```
    def setUp(self):
        desired_caps = {}
        desired_caps['platform'] = self.PLATFORM
        desired_caps['browserName'] = self.BROWSER

        sauce_string = self.SAUCE_USERNAME + ':' +
self.SUACE_KEY

        self.driver = \
            webdriver.Remote('http://' +
sauce_string +
            '@ondemand.saucelabs.com:80/wd/hub',
desired_caps)

self.driver.get('http://demo.magentocommerce.com/'
)
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

    def testSearchByCategory(self):
        # get the search textbox
        self.search_field =
self.driver.find_element_by_name('q')
        self.search_field.clear()

        # enter search keyword and submit
        self.search_field.send_keys('phones')
        self.search_field.submit()

        # get all the anchor elements which have
product names
        # displayed currently on result page using
        # find_elements_by_xpath method
        products = self.driver.\

find_elements_by_xpath('//h2[@class=\'product-
```

```
name\']/a')

        # check count of products shown in results
        self.assertEqual(2, len(products))

    def tearDown(self):
        # close the browser window
        self.driver.quit()

if __name__ == '__main__':
    if len(sys.argv) > 1:
        SearchProducts.BROWSER = sys.argv.pop()
        SearchProducts.PLATFORM = sys.argv.pop()
    unittest.main(verbosity=2)
```

□5□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□

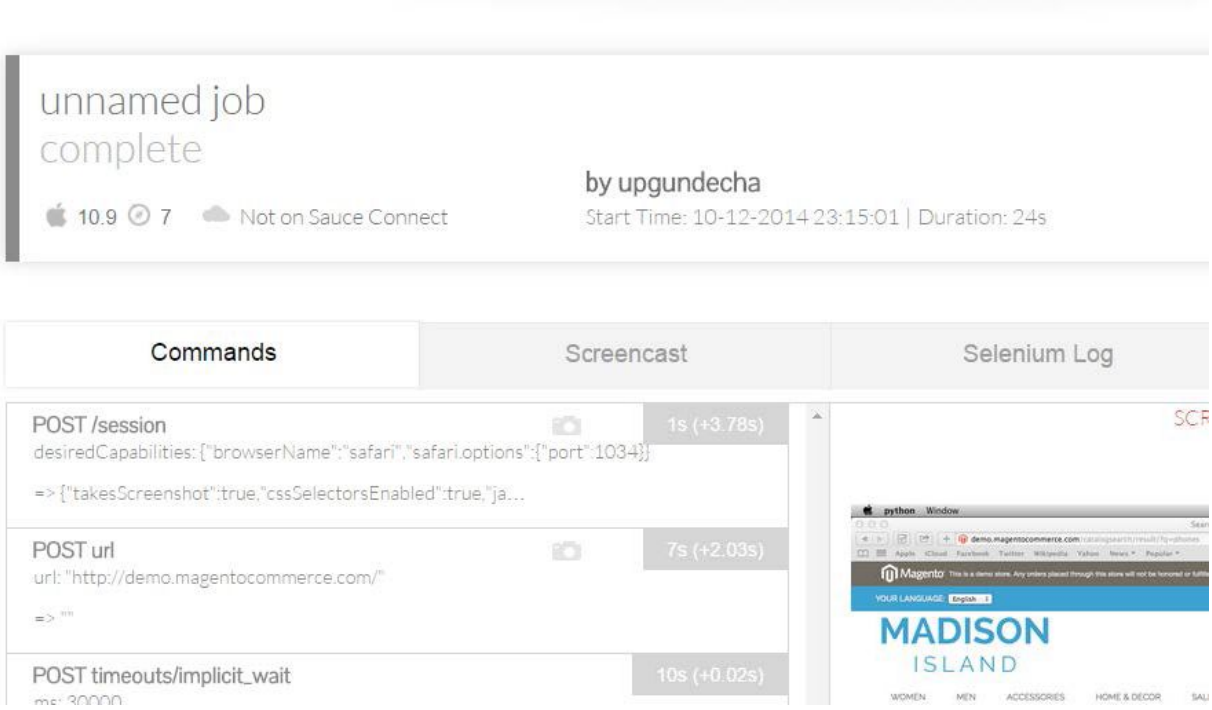
```
python sauce_test.py "OS X 10.9" "Safari"
```

> □□□
> https://saucelabs.com/platforms□□
> Sauce Labs□□□□□□□□

□□□□□□□□□□Sauce Labs□Grid Server□□□□□□□□□□□□□□□□Sauce□□□□□□□□□□□□□□□□□□□□□□□□□□

□6□□□□□□Sauce Dashboard□□□□□□□□□□□□□□

□□□□□□Sauce□□□□□□□□□□□□□□□□□□□□□□□□□□□□Sauce□□□□□□□□□□□□□□□□□□□□Selenium□□□□□□Selenium□□□□□□□□□□□□□□□□□□□□



□□□□Sauce Connect□□□□□□□□□□□□□□□□□□□□Sauce Connect□□□□□□□Sauce□□□□□□□□□□□□

6.7 □□□□

□□□□□□□□□□□□□□□□Selenium Standalone Server□□□□□□□□□□□□Selenium Standalone

Server[illegible]

[illegible]Selenium Grid[illegible]Selenium Grid[illegible]

[illegible]Sauce Labs[illegible]Sauce Labs[illegible]

[illegible]Appium[illegible]Selenium WebDriver[illegible]Appium[illegible]iOS[illegible]Android[illegible]Web[illegible]Appium[illegible]

第7章　自动化测试

自动化测试是软件测试的一个重要组成部分，它可以帮助测试人员快速地执行测试用例，并且可以减少人为错误的发生。在移动应用开发中，自动化测试也是非常重要的，因为移动应用的开发周期通常比较短，需要快速地进行测试和发布。本章将介绍如何使用WebDriver和Appium进行移动应用的自动化测试。

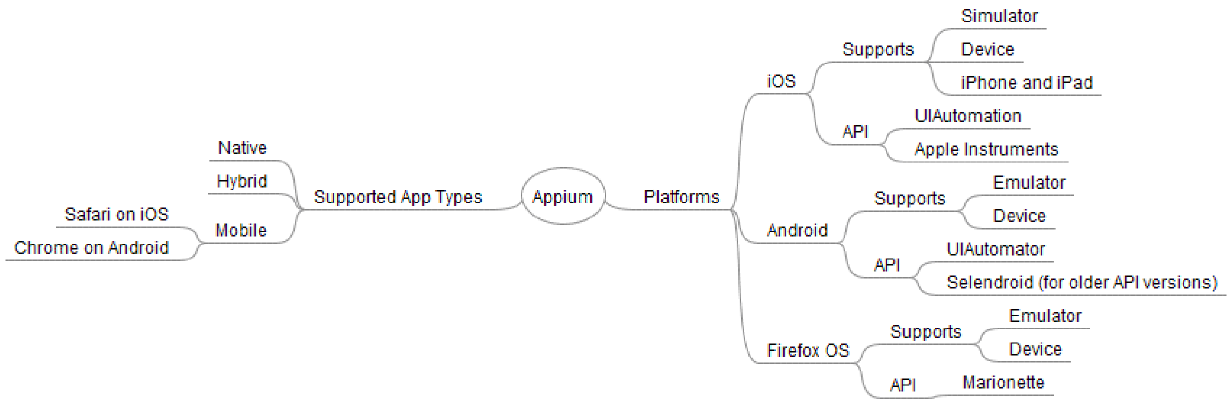
本章学习目标

- 了解Appium的基本概念
- Appium的安装与配置
- 在iPhone模拟器上运行iOS测试
- 在真实设备上运行Android测试

7.1 □□Appium

Appium□□□□□□□□□□□□□□□□□□□□□□□□□□□□iOS□Android□Firefox OS □□□□□□□□□□□□□□□□□□□□Selenium WebDriver□□□□□□□□□□Selenium Server □□□□JSON Wire Protocol□□Selenium 2□□Appium□□□ iPhoneDriver □ AndroidDriver API□□□□□□□□□□□□□□□□□□

Appium□□□□□□□□□□□□□□□□Selenium WebDriver □□□□□□□□□□□□□□Appium□□□ Selenium WebDriver □□□□□□□□□□□□□□□□□□□□ Selenium client library□□□□□□□□□□□□□□□□□□□□ □□□□□□□Appium□□□□□□□□□□□□□□□□□□□□□□□□□□



7.1.1 Appium□□□□□□□

Appium □□□□□□□□□□□□

- □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□iPhone□iPad□□□□□□□□Objective-C□iOS SDK □□□□□□□Android □□□□□Java□Android SDK□□
- □□□**Web**□□□□□Web□□□□□□□□□□□PHP□Java□□ASP.NET□□□□□□□□□□□□□□□jQuery Mobile□Sencha Touch □□□□□□□□□□□□□□□UI□

- [illegible]HTML5[illegible]CSS[illegible]JavaScript[illegible]HTML[illegible]WebView[illegible]JavaScript[illegible]Web[illegible]

7.1.2 Appium[illegible]

[illegible]Appium[illegible]iOS[illegible]Android[illegible]

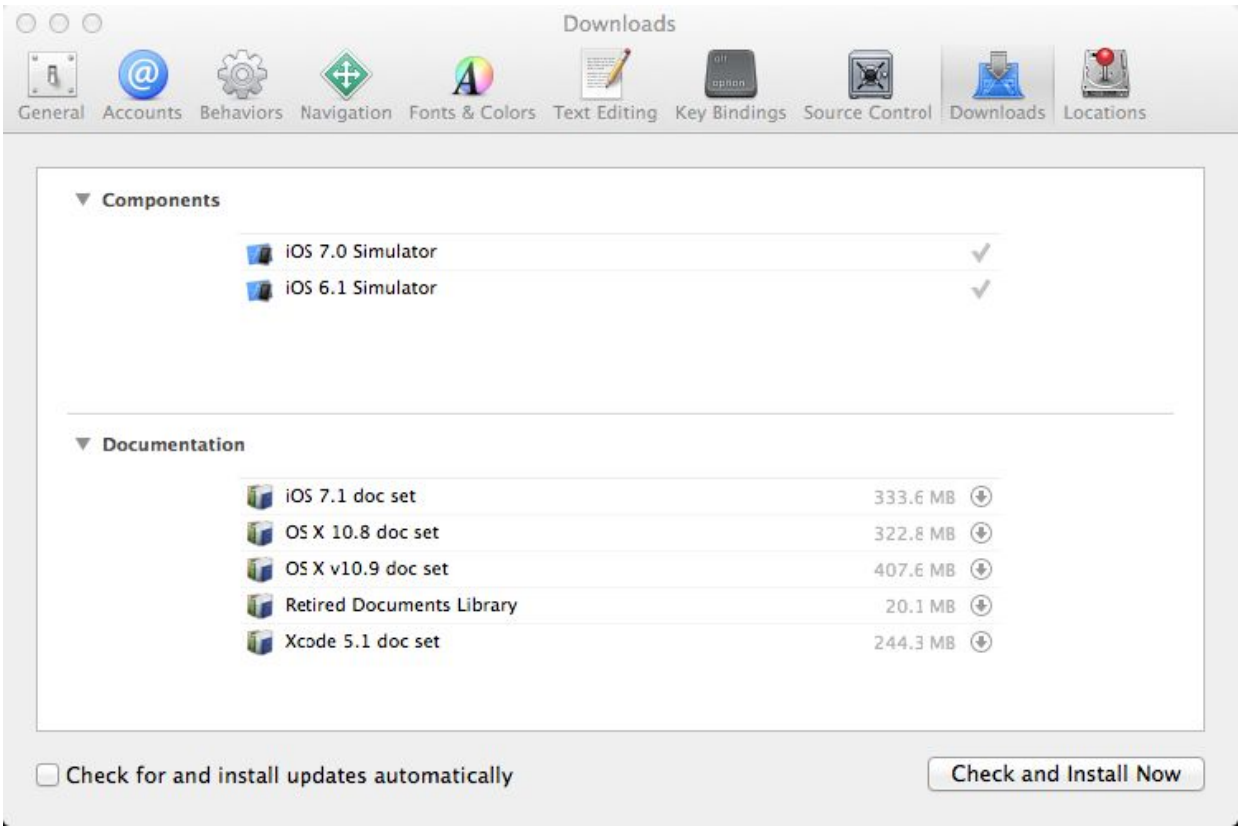
> Appium[illegible]Node.js[illegible]Mac OS X[illegible]Windows[illegible]Standalone GUI[illegible]Node.js [illegible]Mac OS X[illegible]Node.js[illegible]Appium Standalone GUI[illegible]

7.1.2.1 [illegible]Xcode

□□□□□Mac OS X□□□□□Xcode 4.6.3□□□□□
□□□□□□iOS□□□□□□□

□□□□□□□□□Xcode 5.1□□□□□□App Store
□□□□□□□□□□□□□
https://developer.apple.com/xcode/□

□□□Xcode□□□□□□□□□□□□□□□□□□□□
Preferences | Downloads□□□Command Line
Tools□□□□iOS SDK□□□□□□□□□□iOS□□□□□□□
□□□□□□□□□



□□□□□□□□□□□□□□□□□□□□□ Provision Profile□□□□□USB□□□□□

□□iPhone□□□□□□□□□□□□□□□□□□□□□Xcode | Open Developer Tool | iOS Simulator □□□□□□□□□□□□Safari□□□□□□□□□□□□□Web□□□□□□

7.1.2.2 □□Android SDK

□□□□□□Android SDK□□□Android□□□□□□□□□□□□□□□□□□□Android SDK□ http://developer.android.com/sdk/□□□□□□□□□□ANDROID_HOME□□□□□□□□□□□□□□□path□□

□□□□□□□□□□ http://developer.android.com/sdk/installing/index.html?pkg=tools□

□□□□□Appium□□□□□□□□□□□□□□□□□□□□□

iOS Simulator - iPhone Retina (4-inch 64-bit) / iOS 7.1 (11D167)

Carrier 10:40 PM

demo.magentocommerce.com

Magento This is a demo store. Any orders placed through this store will not be honored or fulfilled.

MADISON ISLAND English

Menu

HOME & DECOR
FOR ALL YOUR SPACES

SHOP PRIVATE SALES

7.1.2.3 □□Appium Python client

□□□□□□□Appium Python client□□□□□ Selenium 3.0□□□□□□□□□□□□□□□□□□□Python□ Appium□□□□□□□□□□□□□□□□□□□□□□□□Appium Python client□

```
pip install Appium-Python-Client
```

□□□□□□□□□□□□□□□Appium Python client □□□□□□□ https://pypi.python.org/pypi/Appium-Python-Client□

7.2 □□Appium

[illegible]Appium[illegible]

Appium[illegible]Appium GUI[illegible]iPhone[illegible]iPad[illegible]iOS[illegible]Mac OS X[illegible]Appium[illegible]Android[illegible]Windows[illegible]Linux[illegible]Appium[illegible]Mac OS X[illegible]Appium[illegible]Appium[illegible]http://appium.io/[illegible]

[illegible]

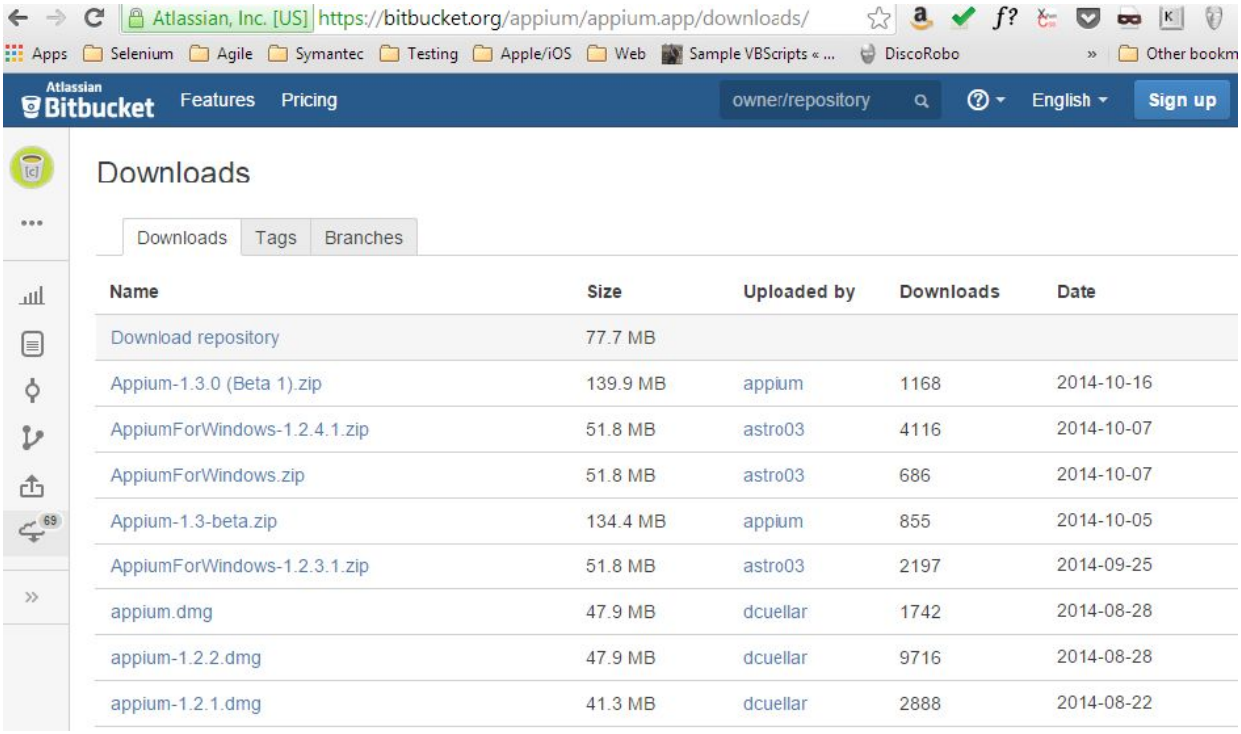
[illegible]1[illegible]**Download Appium**[illegible]

□2□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□

> □□□□□□□□□□□□Mac OS X □□□□□
> Appium□

□3□□Mac□□□□□□□□□□□□□□□□□Appium□□□□Appium□□□Applications□□□□



□□□□□Applications□□□□Appium□□□□□□□□□□□□□□□□□iOS□□□□

□□□□□□Appium□□□□URL□□□□ http://127.0.0.1:4723□localhost□□URL□□□□□□□□□□□□□□□□□□□□□□□□□□□iPhone□ Safari□□□□□□□□□□□□□□□□□□□□□□□

□4□□□□□□□□Appium□□□□□□□Apple□□□□□iOS□□□□□□

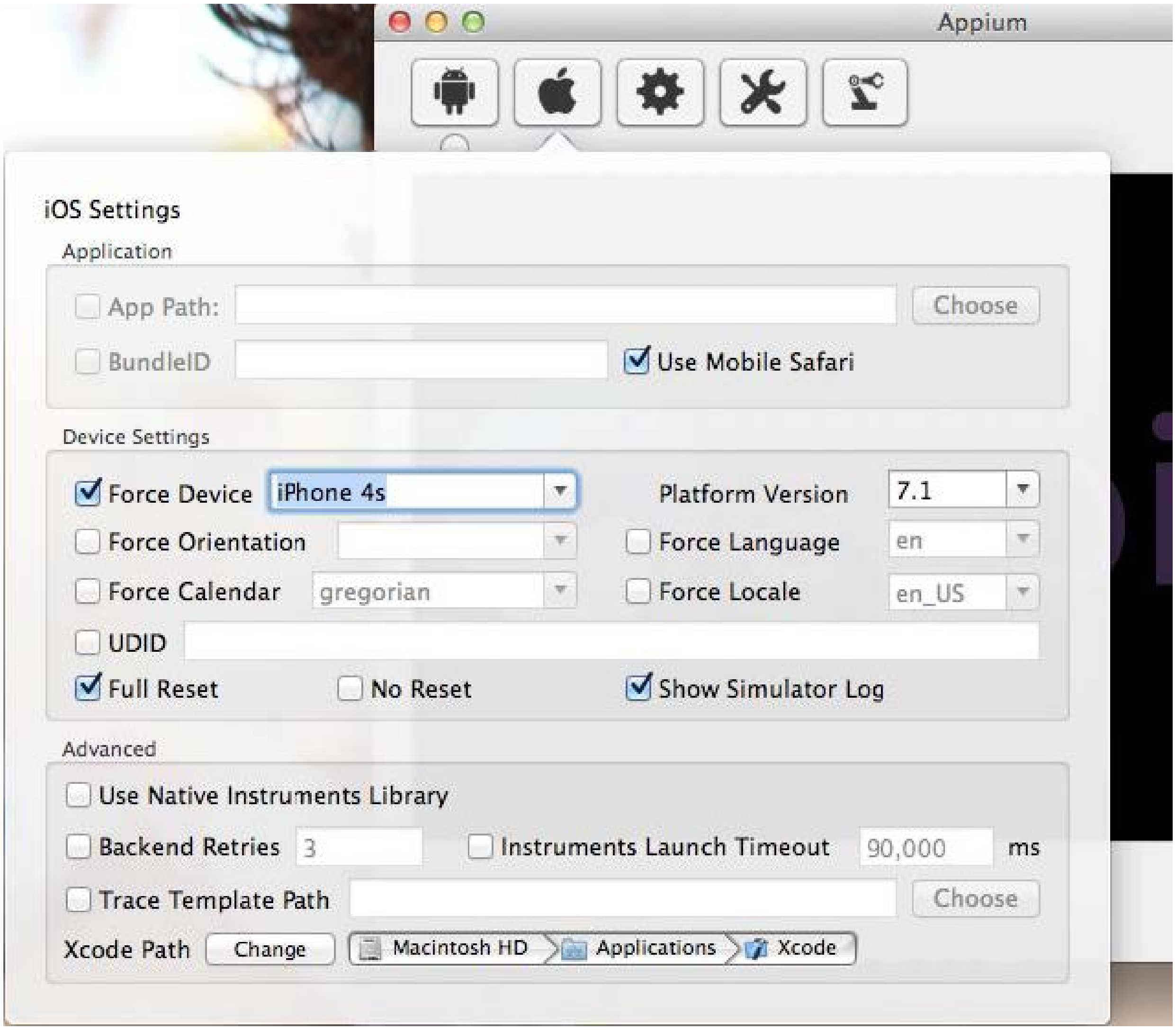
□5□□iOS□□□□□□□□□Force Device□□□□□□□□□□□□□□iPhone□□□□□□□□Use Mobile Safari□□□□□□□□□□

□6□□Appium□□□□□□Launch□□□□Appium Server□

Inspector□□□□□□Appium□□□□□□□□Appium Inspector□□□□□□Appium□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□UI□□

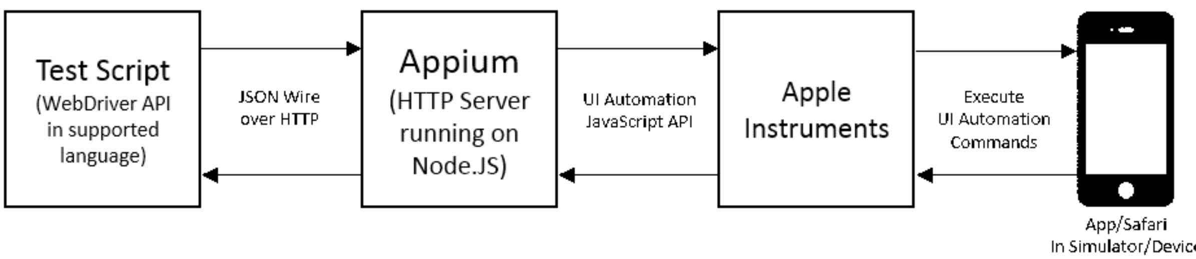
Appium
Launch
appium



7.3 iOS□□

Appium□□□□□□□□□□□□□□□□□□□□□□□□□□□Selenium WebDriver JSON□□□□□API□□□□□iOS□□□□□□□□□□□Appium□□□Apple□□□□□UI Automation □□□□□□□

Appium [illegible]HTTP Server[illegible]JSON Wire Protocol[illegible]Appium[illegible]Apple[illegible]Appium[illegible]JSON[illegible]UI Automation [illegible]JavaScript[illegible]UI Automaiton[illegible]



[illegible]UI Automation[illegible]JavaScript[illegible]Appium[illegible]Appium[illegible]UI Automation JavaScript[illegible]Selenium WebDriver JSON Wire Protocol[illegible]

□□□□□□□□□□□□Appium□□□□□□□□□□
iPhone Safari□□□□□□Web□□□□□□□□□□□□□□□□
□□□□□□SearchProductsOnIPhone□□□□□□

```
import unittest
from appium import webdriver

class SearchProductsOnIPhone(unittest.TestCase):
    def setUp(self):
        desired_caps = {}
        # platform
        desired_caps['device'] = 'iPhone Simulator'
        # platform version
        desired_caps['version'] = '7.1'
        # mobile browser
        desired_caps['app'] = 'safari'

        # to connect to Appium server use
RemoteWebDriver
        # and pass desired capabilities
        self.driver = \

webdriver.Remote("http://127.0.0.1:4723/wd/hub",
desired_caps)

self.driver.get("http://demo.magentocommerce.com/"
)
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

    def test_search_by_category(self):
        # click on search icon
        self.driver.find_element_by_xpath
        ("//a[@href='#header-search']").click()
```

```
        # get the search textbox
        self.search_field =
self.driver.find_element_by_name("q")
        self.search_field.clear()
        # enter search keyword and submit
        self.search_field.send_keys("phones")
        self.search_field.submit()

        # get all the anchor elements which have
product names
        # displayed currently on result page using
        # find_elements_by_xpath method
        products = self.driver\
            .find_elements_by_xpath
            ("//div[@class='category-
products']/ul/li")

        # check count of products shown in results
        self.assertEqual(2, len(products))

    def tearDown(self):
        # close the browser window
        self.driver.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

□□□□RemoteWebDriver □□□Appium□□□□□□□□□□□□□□□□□□□□Appium□□□□□□□□□□

```
desired_caps = {}
# platform
desired_caps['device'] = 'iPhone Simulator'
# platform version
```

```
desired_caps['version'] = '7.1'
# mobile browser
desired_caps['app'] = 'safari'
```

desired_caps['device'] [illegible]Appium[illegible]iPhone [illegible]iPad[illegible]iPad [illegible]

[illegible]desired_caps['device'][illegible]iPhone[illegible]iPad[illegible]Appium[illegible]USB[illegible]Mac[illegible]

desired_caps['version'][illegible]iPhone/iPad[illegible]iOS 7.1[illegible]iOS[illegible]

desired_caps['app'][illegible]Safari[illegible]

[illegible]RemoteWebDriver[illegible]Appium Server[illegible]Remote [illegible]

□□

```
self.driver = webdriver.Remote
  ("http://127.0.0.1:4723/wd/hub", desired_caps)
```

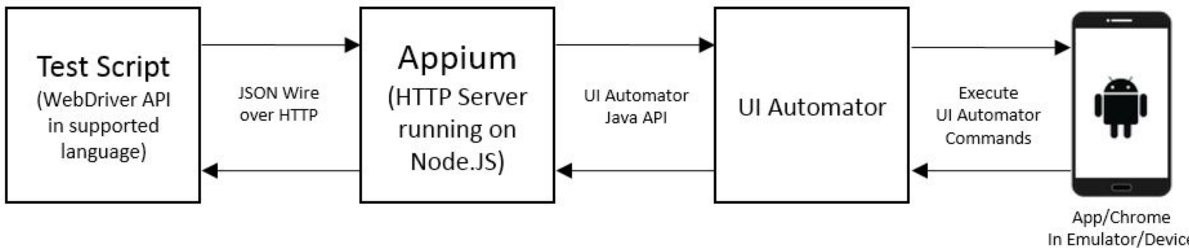
□□□□□□□□□□□Selenium API□□□□□□□□□□□□□□□□□□□□□□□□Appium□□□□□□□□□□□□□□ iPhone□□□□□□Safari□□□□□Appium□□□□□□□□□□□□□□□□□Safari□□□□□□

7.4 Android□□

Appium□□Android□□□□□□□□□□□□□□□□□□ Android SDK□□ UI Automator□□□□□□□□□□□□□□□□iOS□□□□□□

Appium □□□□HTTP Server□□□□□□□□□ JSON Wire Protocol□□□□□□□□Appium□□□□□□UI Automator□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□Appium□JSON□□□□□□UI Automator □□□□□□Java□□□□□□□□□□□□□□□



□□□UI Automator□□□□□□□Appium□Appium□□UI Automator□□□□□□□□Selenium WebDriver JSON Wire Protocol□□□□□□□□□□□□□□□□□□□□

□Android□□□□□□□□□□□□□□□iOS□□□□□□□□□□□□□□□□□Android□□□□□□□□□□□□□□□□□□□□□iOS□□□simulator□□□□□Android□□□□□□emulator□□□□□□□□□□□□□□□□□□Android□□□□□Chrome□□□□□

□□□□□□□□□□□□Samsung Galaxy S III□□□□□□□□□□□□□□□□□□□□Chrome□□□□□Chrome□□□□□□□□

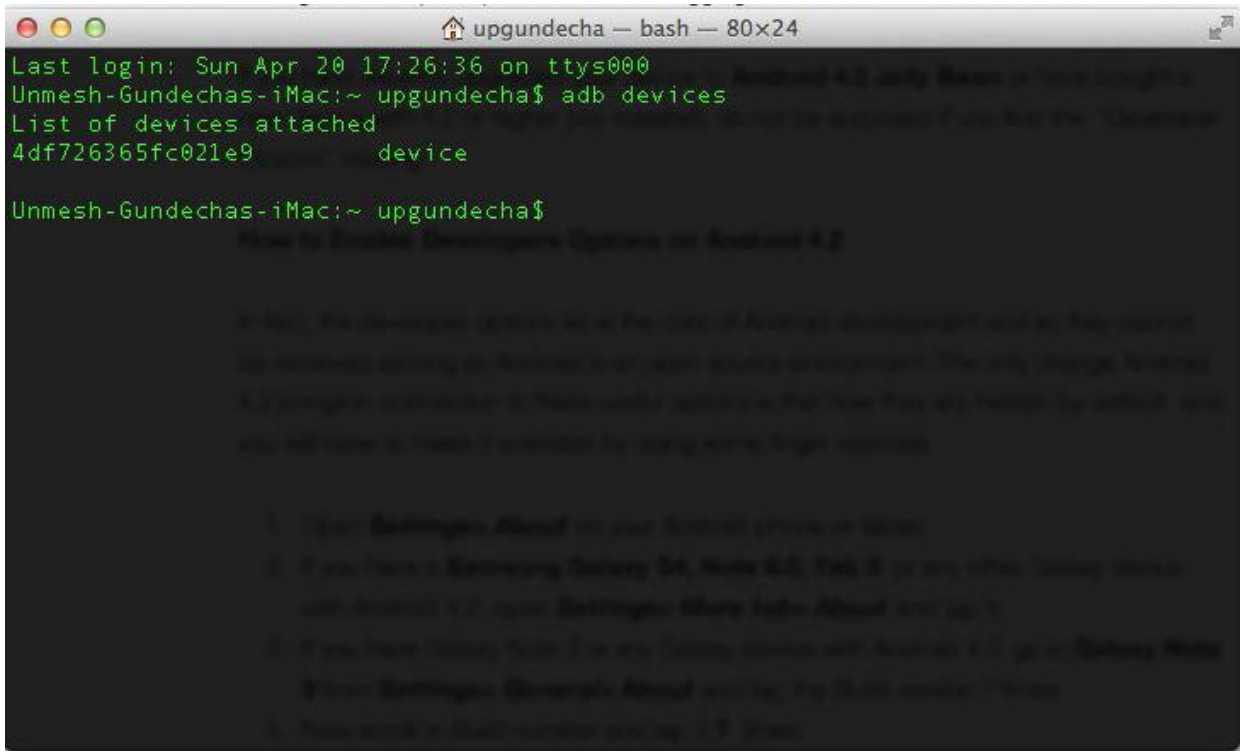
[illegible]Appium[illegible]

[illegible]Android[illegible]Android[illegible]Chrome[illegible]

```
./adb devices
```

Android Debug Bridge(adb) [illegible]Android SDK[illegible]

[illegible]Android[illegible]



□□□□□□□□□iOS□□□□□□□□□□□□□□Android□□□□□□□□□□□□□□□SearchProductsOnAndroid□□□□□□□□□□□□□□□□□□□□□

```
import unittest
from appium import webdriver

class SearchProductsOnAndroid(unittest.TestCase):
    def setUp(self):
        desired_caps = {}
        # platform
        desired_caps['device'] = 'Android'
        # platform version
        desired_caps['version'] = '4.3'
        # mobile browser
```

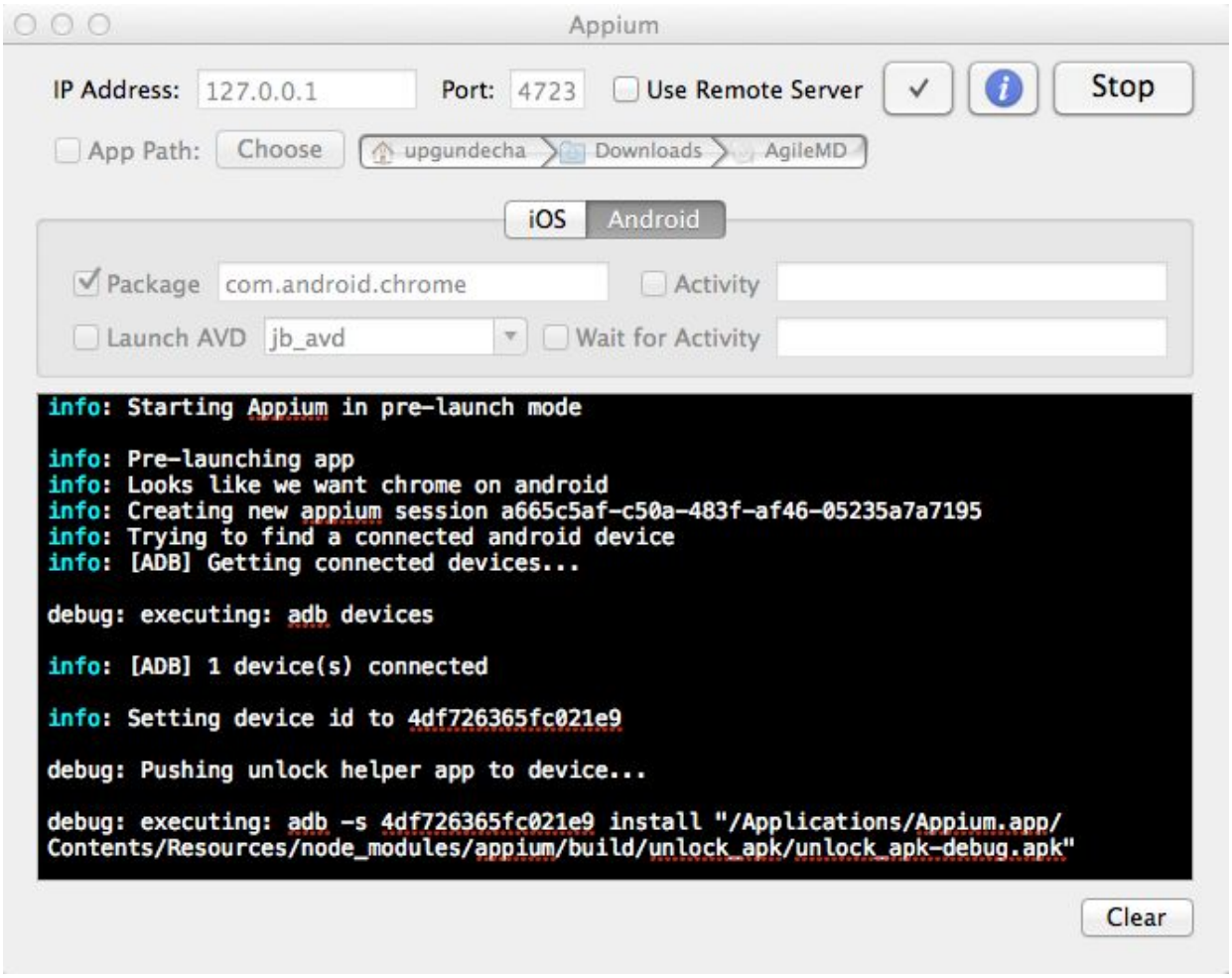
```
        desired_caps['app'] = 'Chrome'

        # to connect to Appium server use
RemoteWebDriver
        # and pass desired capabilities
        self.driver = \

webdriver.Remote("http://127.0.0.1:4723/wd/hub",
desired_caps)

self.driver.get("http://demo.magentocommerce.com/"
)
        self.driver.implicitly_wait(30)

    def test_search_by_category(self):

        # click on search icon
        self.driver.find_element_by_xpath
          ("//a[@href='#header-search']").click()
        # get the search textbox
        self.search_field =
self.driver.find_element_by_name("q")
        self.search_field.clear()

        # enter search keyword and submit
        self.search_field.send_keys("phones")
        self.search_field.submit()

        # get all the anchor elements which have
product names
        # displayed currently on result page using
        # find_elements_by_xpath method
        products = self.driver\
          .find_elements_by_xpath
            ("//div[@class='category-
products']/ul/li")
```

```
        # check count of products shown in results
        self.assertEqual(2, len(products))

    def tearDown(self):
        # close the browser window
        self.driver.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

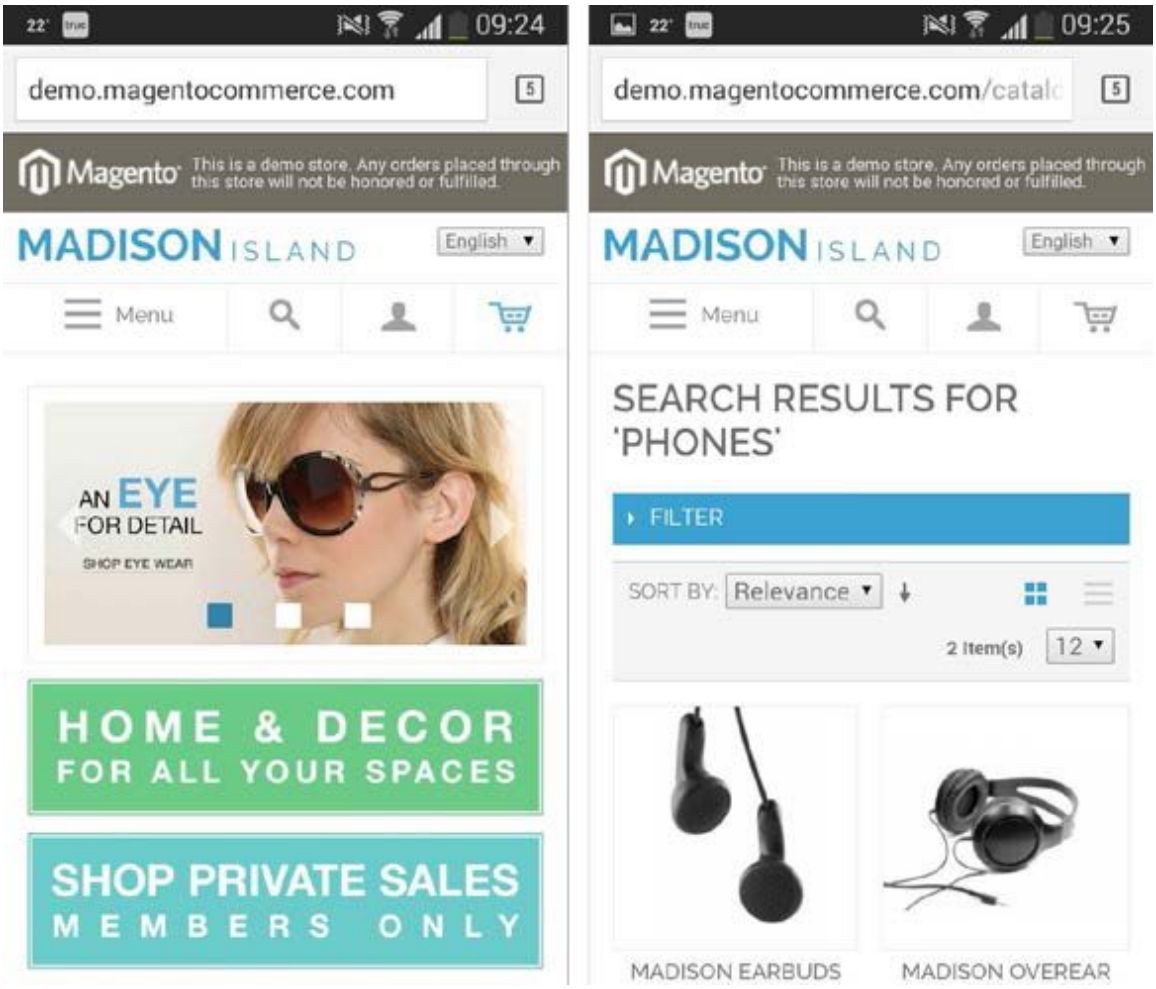
□□□□□□□□□desired_caps['device']□□□□□Android□□□□□□Android□□□□□□□□

□□□□□□□□□desired_caps['version']□□□□Android 4.3□□□Jelly Bean□□□□□Android□□□Chrome□□□□□□□□□□□□desired_caps['app']□□□□□Chrome□

Appium□□□□□□adb□□□Chrome□□□□□□□□□□□□□□□□□□□□□□


Appium
IP Address: 127.0.0.1
Port: 4723
Use Remote Server
Stop
App Path:
Choose
upgundecha
Downloads
AgileMD
iOS
Android
Package
com.android.chrome
Activity
Launch AVD
jb_avd
Wait for Activity
info: Starting Appium in pre-launch mode
info: Pre-launching app
info: Looks like we want chrome on android
info: Creating new appium session a665c5af-c50a-483f-af46-05235a7a7195
info: Trying to find a connected android device
info: [ADB] Getting connected devices...
debug: executing: adb devices
info: [ADB] 1 device(s) connected
info: Setting device id to 4df726365fc021e9
debug: Pushing unlock helper app to device...
debug: executing: adb -s 4df726365fc021e9 install "/Applications/Appium.app/
Contents/Resources/node_modules/appium/build/unlock_apk/unlock_apk-debug.apk"
Clear

□□□□□□□□□□□□□□□□□□□



7.5 □□Sauce Labs

□□6□□□□□□□□□□□□□□□□Sauce Labs□Sauce□□□□□Appium□□□□□□□□□□□□□□□Appium□□□□□Sauce Labs□□□□□□□□□□□□□□□□□□□□□□□□□□□Sauce Labs□□□□□□□□□□□□□□□□□□□□□□

```
import unittest
from appium import webdriver

class SearchProductsOnIPhone(unittest.TestCase):
   SAUCE_USERNAME = 'upgundecha'
   SUACE_KEY = 'c6e7132c-ae27-4217-b6fa-
3cf7df0a7281'

   def setUp(self):
      desired_caps = {}
      desired_caps['browserName'] = "Safari"
      desired_caps['platformVersion'] = "7.1"
      desired_caps['platformName'] = "iOS"
      desired_caps['deviceName'] = "iPhone
Simulator"

      sauce_string = self.SAUCE_USERNAME + ':' +
self.SUACE_KEY

      self.driver = \
         webdriver.Remote('http://' +
sauce_string +
          '@ondemand.saucelabs.com:80/wd/hub',
desired_caps)

self.driver.get('http://demo.magentocommerce.com/'
)
      self.driver.implicitly_wait(30)
      self.driver.maximize_window()

   def test_search_by_category(self):
      # click on search icon

self.driver.find_element_by_xpath("//a[@href=
      '#header-search']").click()
      # get the search textbox
      self.search_field =
```

```
self.driver.find_element_by_name("q")
        self.search_field.clear()

        # enter search keyword and submit
        self.search_field.send_keys("phones")
        self.search_field.submit()

        # get all the anchor elements which have
        # product names displayed
        # currently on result page using
        # find_elements_by_xpath method
        products = self.driver\
            .find_elements_by_xpath
             ("//div[@class='category-
products']/ul/li")

        # check count of products shown in results
        self.assertEqual(2, len(products))

    def tearDown(self):
        # close the browser window
        self.driver.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

□□□□□□□□□□□□□□□□Sauce Labs□□□□□□□□□□□□□□□□Sauce Labs□□□□□□□SDK□□□□□□□□Sauce Labs□□□□□□□□□□□□□□□□□□□□Appium□□□□□□

7.6 [illegible]

[illegible]Appium[illegible]Selenium[illegible]Appium[illegible]

[illegible]iPhone[illegible]Android[illegible]Appium[illegible]WebDriver[illegible]

[illegible]Selenium WebDriver[illegible]Page Object[illegible]

第8章 Page Object设计模式

在前面的章节中，我们已经学习了Python编程语言和Selenium自动化测试工具的基本使用方法。

在本章中，我们将介绍Page Object设计模式，这是一种在自动化测试中广泛使用的设计模式，它可以帮助我们更好地组织和管理测试代码。

本章学习目标

- 理解设计模式
- 掌握使用ddt数据驱动unittest框架的方法

- [illegible]Page Object[illegible]
- [illegible]Page Object[illegible]

8.1 [illegible]

[illegible]

[illegible]“[illegible]”[illegible]

[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]

[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]

[illegible]

[illegible]CSV[illegible]

[illegible]

[illegible]Python ddt[illegible]

8.2 [illegible]ddt[illegible]

ddt[illegible]

[illegible]

ddt[illegible]

8.2.1 [illegible]ddt

[illegible]ddt[illegible]

```
pip install ddt
```

[illegible]ddt[illegible]

https://pypi.python.org/pypi/ddt[illegible]

8.2.2 [illegible]

[illegible]

[illegible]

[illegible]@ddt[illegible]

[illegible]@data[illegible]@data[illegible]

□□□□□□□□□□□□□□□□□□□□□□□□□□□□@unpack□□□□
□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

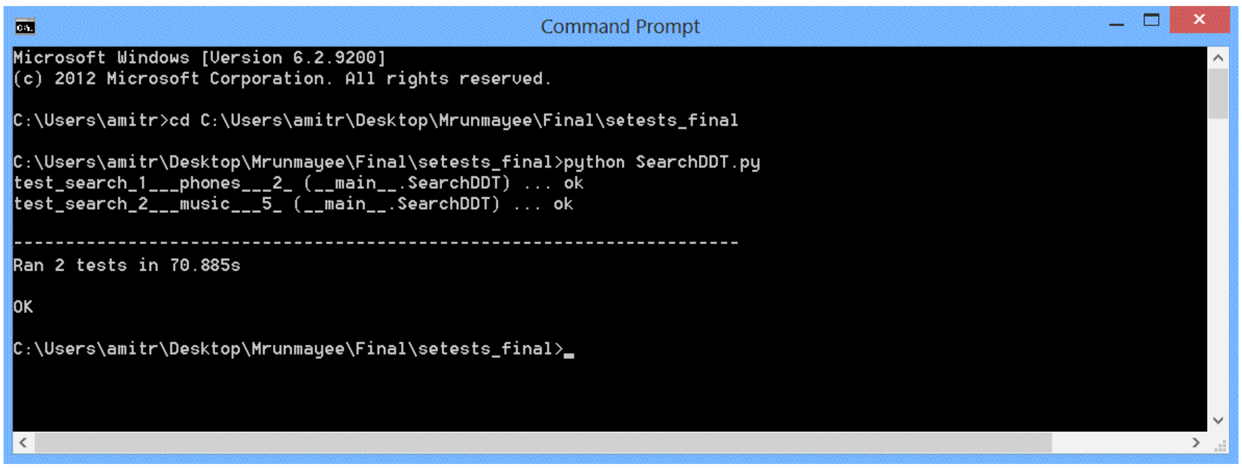
```
import unittest
from ddt import ddt, data, unpack
from selenium import webdriver

@ddt
class SearchDDT(unittest.TestCase):
    def setUp(self):
        # create a new Firefox session
        self.driver = webdriver.Firefox()
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

        # navigate to the application home page

self.driver.get("http://demo.magentocommerce.com/"
)

    # specify test data using @data decorator
    @data(("phones", 2), ("music", 5))
    @unpack
    def test_search(self, search_value,
expected_count):
        # get the search textbox
        self.search_field =
self.driver.find_element_by_name("q")
        self.search_field.clear()
        # enter search keyword and submit.
        # use search_value argument to pass data
```

```
        self.search_field.send_keys(search_value)
        self.search_field.submit()

        # get all the anchor elements which have
        # product names displayed
        # currently on result page using
        # find_elements_by_xpath method
        products =
self.driver.find_elements_by_xpath
          ("//h2[@class='product-name']/a")

        # check count of products shown in results
        self.assertEqual(expected_count,
len(products))

    def tearDown(self):
        # close the browser window
        self.driver.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

□□□□□□□□□□@data□□□□□□□□□□□□□□□□□□@unpack□□□□□□□□□□□□□□□□□□ test_search()□□□□search_value □ expected_count□□□□□□□□□□□□□□□□□□

```
# specify test data using @data decorator
    @data(("phones", 2), ("music", 5))
    @unpack
    def test_search(self, search_value,
expected_count):
```

[illegible]ddt[illegible]Pyhotn [illegible]ddt[illegible]



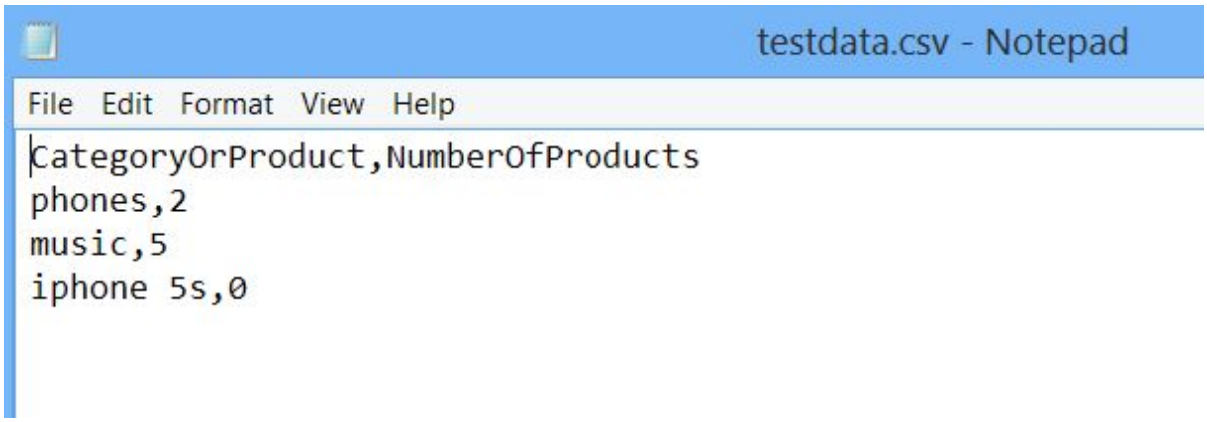
8.3 [illegible]

[illegible]

[illegible]CSV[illegible] Excle[illegible]ddt[illegible]

8.3.1 □□CSV□□□□

□□□□□□□□□□□□□□@data□□□□□□□□□□□□CSV□testdata.csv□□□□□□□□□□□□□□□CSV□□□□□□□□□□□



□□□□□□□□@data□□□□□□get_data()□□□□□□□□□□CSV□□□□□□□□□□CSV□□□□□□□□□□□□□□□

```
import csv, unittest
from ddt import ddt, data, unpack
from selenium import webdriver

def get_data(file_name):
    # create an empty list to store rows
    rows = []
    # open the CSV file
```

```
    data_file = open(file_name, "rb")
    # create a CSV Reader from CSV file
    reader = csv.reader(data_file)
    # skip the headers
    next(reader, None)
    # add rows from reader to list
    for row in reader:
        rows.append(row)
    return rows

@ddt
class SearchCsvDDT(unittest.TestCase):
    def setUp(self):
        # create a new Firefox session
        self.driver = webdriver.Firefox()
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

        # navigate to the application home page

self.driver.get("http://demo.magentocommerce.com/"
)

    # get the data from specified csv file by
    # calling the get_data function
    @data(*get_data("testdata.csv"))
    @unpack
    def test_search(self, search_value,
expected_count):
            self.search_field
=self.driver.find_element_by_name("q")
            self.search_field.clear()

            # enter search keyword and submit.

self.search_field.send_keys(search_value)
            self.search_field.submit()
```

```
            # get all the anchor elements which have
            # product names displayed
            # currently on result page using
            # find_elements_by_xpath method
            products =
self.driver.find_elements_by_xpath
              ("//h2[@class='product-name']/a")
            expected_count = int(expected_count)
            if expected_count > 0:
                # check count of products shown in
results
                self.assertEqual(expected_count,
len(products))
            else:
                msg =
self.driver.find_element_by_class_name ("note-
msg")
                self.assertEqual ("Your search
returns no results.", msg.text)

    def tearDown(self):
        # close the browser window
        self.driver.quit()

if __name__ == '__main__':
    unittest.main()
```

□□□□□□@data□□□get_data()□□□□□□□□□

□□□□□□□□□□□□□@data□

8.3.2 □□Excel□□□□

[illegible]Excel[illegible]
[illegible]Excel
[illegible]

A	B
Category/Product	NumberOfProducts
phones	2
music	5
iphone 5s	0

[illegible]Excel[illegible]xlrd[illegible]

```
pip install xlrd
```

xlrd[illegible]xlwt[illegible]openpyxl[illegible]

□□□□□□□get_data()□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
import xlrd, unittest
from ddt import ddt, data, unpack
from selenium import webdriver

def get_data(file_name):
    # create an empty list to store rows
    rows = []
    # open the specified Excel spreadsheet as
workbook
    book = xlrd.open_workbook(file_name)
    # get the first sheet
    sheet = book.sheet_by_index(0)
    # iterate through the sheet and get data from
rows in list
    for row_idx in range(1, sheet.nrows):
        rows.append(list(sheet.row_values(row_idx,
0, sheet.ncols)))
    return rows

@ddt
class SearchExcelDDT(unittest.TestCase):
    def setUp(self):
        # create a new Firefox session
        self.driver = webdriver.Firefox()
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

        # navigate to the application home page

self.driver.get("http://demo.magentocommerce.com/"
)
```

```
   # get the data from specified Excel
spreadsheet
   # by calling the get_data function
   @data(*get_data("TestData.xlsx"))
   @unpack
   def test_search(self, search_value,
expected_count):
         self.search_field =
self.driver.find_element_by_name("q")
           self.search_field.clear()

         # enter search keyword and submit.

self.search_field.send_keys(search_value)
         self.search_field.submit()

         # get all the anchor elements which have
         # product names displayed
         # currently on result page using
         # find_elements_by_xpath method
         products =
self.driver.find_elements_by_xpath
            ("//h2[@class='product-name']/a")
         if expected_count > 0:
            # check count of products shown in
results
            self.assertEqual(expected_count,
len(products))
         else:
            msg = self.driver.
find_element_by_class_name("note-msg")
            self.assertEqual("Your search
returns no results.", msg.text)

   def tearDown(self):
      # close the browser window
      self.driver.quit()
```

```
if __name__ == '__main__':
    unittest.main()
```

□□□□CSV□□□□□@data□□□get_data()□□□□□□□Excel□□□□□□□□□□□□@data□

□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□get_data()□□□□□□□DB□□□□□□□□□□□SQL□□□□□□□□□□

8.4 Page Object□□□□

□□□□□□□□□□□□□□Python unittest□□□□□□□□□□Selenium WebDriver□□□

[illegible]

[illegible]

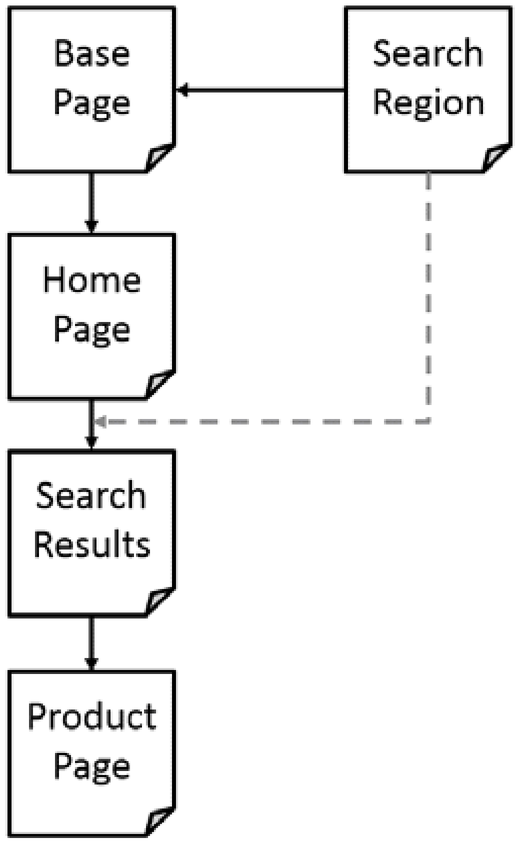
[illegible]Don't Repeat Yourself[illegible]DRY[illegible]Page Object[illegible]Selenium[illegible]Facade[illegible]

Page Object[illegible]Page Object[illegible]API[illegible]

[illegible]Page Object[illegible]

- [illegible]
- [illegible]
- [illegible]

[illegible]Base Page [illegible]Base [illegible]Base Page[illegible]Search[illegible]



8.4.1 □□□□

□□□□□□Page Object□□□□□□□□□□□□□□□□□□□BaseTestCase□□□□□□□□□□setUp()□tearDown()□□□□□□□□□□□□□□□□□□□□□□□□□□□basetestcase.py□□□□□□□□□□□

```
import unittest
from selenium import webdriver

class BaseTestCase(unittest.TestCase):
```

```
    def setUp(self):
        # create a new Firefox session
        self.driver = webdriver.Firefox()
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

        # navigate to the application home page

self.driver.get('http://demo.magentocommerce.com/'
)

    def tearDown(self):
        # close the browser window
        self.driver.quit()
```

8.4.2 BasePage[illegible]

BasePage [illegible]

[illegible]base.py[illegible]

```
from abc import abstractmethod
class BasePage(object):
    """ All page objects inherit from this """

    def __init__(self, driver):
        self._validate_page(driver)
        self.driver = driver

    @abstractmethod
    def _validate_page(self, driver):
        return

    """ Regions define functionality available
```

```
through all page objects """
    @property
    def search(self):
        from search import SearchRegion
        return SearchRegion(self.driver)

class InvalidPageException(Exception):
    """ Throw this exception when you don't find
the correct page """
    pass
```

□□□□□□□□□□_validate_page()□□□□□□□□□BasePage□page□□

□□□□□□□□□□search□□□□□□□SearchRegion□□□□□□□□□□□□□□SearchRegion□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□BasePage□□

□□□□□□□□□□_validate_page()□□□□□□□InvalidPageException□□□□□□□□□□□InvalidPageException□□□□□□□

8.4.3 □□Page Object

□□□□□□□□□□□□□□Page Object□□

□1□□□□□□□□HomePage□□□

homepage.py□□□HomePage□□□□□□□

```
from base import BasePage
from base import InvalidPageException

class HomePage(BasePage):

    _home_page_slideshow_locator = 'div.slideshow-
container'

    def __init__(self, driver):
        super(HomePage, self).__init__(driver)

    def _validate_page(self, driver):
        try:
            driver.find_element_by_class_name
(self._home_page_slideshow_locator)
        except:
            raise InvalidPageException ("Home Page
not loaded")
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□“_”□□□□□□□□□□□□

_home_page_slideshow_locator□□□□□□□□□□

□□slideshow□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
_home_page_slideshow_locator = 'div.slideshow-container'
```

□□□□□□□□□_validate_page()□□□□□□□slideshow□□□□□□□□□□□□□□□□□□□□□□□□

□2□□□□□□□□□SearchRegion □□□□searchFor()□□□□□□□□□□SearchResults□□□□□□□□□□□□□□□□□□□search.py□□□□□□□□□□□□□□□

```
from base import BasePage
from base import InvalidPageException
from product import ProductPage

class SearchRegion(BasePage):
    _search_box_locator = 'q'

    def __init__(self, driver):
        super(SearchRegion, self).__init__(driver)

    def searchFor(self, term):
        self.search_field = 
self.driver.find_element_by_name
```

```
(self._search_box_locator)
        self.search_field.clear()
        self.search_field.send_keys(term)
        self.search_field.submit()
        return SearchResults(self.driver)

class SearchResults(BasePage):
    _product_list_locator = 'ul.products-grid >
li'
    _product_name_locator = 'h2.product-name a'
    _product_image_link = 'a.product-image'
    _page_title_locator = 'div.page-title'

    _products_count = 0
    _products = {}

    def __init__(self, driver):
        super(SearchResults, self).__init__(driver)
        results =
self.driver.find_elements_by_css_selector
(self._product_list_locator)
        for product in results:
            name =
product.find_element_by_css_selector
(self._product_name_locator).text
        self._products[name] =
product.find_element_by_css_selector
(self._product_image_link)

def _validate_page(self, driver):
    if 'Search results for' not in driver.title:
        raise InvalidPageException ('Search results
not loaded')

@property
def product_count(self):
    return len(self._products)
```

```
def get_products(self):
    return self._products

def open_product_page(self, product_name):
    self._products[product_name].click()
    return ProductPage(self.driver)
```

[illegible]3[illegible]ProductPage[illegible] [illegible] SearchResults[illegible] product.py[illegible]ProductPage[illegible]

```
from base import BasePage
from base import InvalidPageException

class ProductPage(BasePage):
    _product_view_locator         = 'div.product-
view'
    _product_name_locator          = 'div.product-
name span'
    _product_description_locator    = 'div.tab-
content div.std'
    _product_stock_status_locator   =
'p.availability span.value'
    _product_price_locator         = 'span.price'

    def __init__(self, driver):
        super(ProductPage, self).__init__(driver)

    @property
    def name(self):
```

```
        return self.driver.\
            find_element_by_css_selector
              (self._product_name_locator)\
            .text.strip()

    @property
    def description(self):
        return self.driver.\
            find_element_by_css_selector
              (self._product_description_locator)\
            .text.strip()

    @property
    def stock_status(self):
        return self.driver.\
            find_element_by_css_selector
              (self._product_stock_status_locator)\
            .text.strip()

    @property
    def price(self):
        return self.driver.\
            find_element_by_css_selector
              (self._product_price_locator)\
            .text.strip()

    def _validate_page(self, driver):
        try:
            driver.find_element_by_css_selector
(self._product_view_locator)
        except:
            raise InvalidPageException ('Product
page not loaded')
```

[illegible]

8.4.4 [illegible]Page Object[illegible]

[illegible]BaseTestCase[illegible]HomePage[illegible]searchFor()[illegible]SearchResults[illegible]SearchResults[illegible]open_product_page()[illegible]

[illegible]searchtest.py[illegible]SearchProductTest[illegible]

```
import unittest
from homepage import HomePage
from BaseTestCase import BaseTestCase

class SearchProductTest(BaseTestCase):
    def testSearchForProduct(self):
        homepage = HomePage(self.driver)
        search_results =
homepage.search.searchFor('earphones')
```

```
        self.assertEqual(2,
search_results.product_count)
        product = search_results.open_product_page
('MADISON EARBUDS')
        self.assertEqual('MADISON EARBUDS',
product.name)
        self.assertEqual('$35.00', product.price)
        self.assertEqual('IN STOCK',
product.stock_status)

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

[illegible]setUp()[illegible]tearDown()[illegible]BaseTestCase[illegible]

[illegible]Page Object[illegible]

8.5 [illegible]

[illegible]Page Object[illegible]

[illegible]unittest[illegible]ddt[illegible]Page Object[illegible]

[illegible]Selenium WebDriver API[illegible]cookies[illegible]

□9□ Selenium WebDriver□□□□□

□□□□□□□□□□□□Selenium WebDriver□□Web□□□□□□
WebDriver□□□□□□□□□□□□□□

□□□□□□□□WebDriver□□□□API□□□□□□□□□□□□□□□□
□□

□□□□□□□

- □□Action□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□
- □□JavaScript□
- □□□□□□
- □□□□□cookies□
- □□□□□□□□

9.1 □□□□□□□

WebDriver□□□□□API□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□WebDriver□Python API□ActionChains□□□□□

□□□□ActionChains□□□□□□□□□□□□□□□□□□□□

□ □	□ □	□ □	□ □
click(on_element=None)	□□□□	on_element:	click(main_link)

	[illegible]	[illegible]None[illegible]	
click_and_hold(on_element=None)	[illegible]	on_element: [illegible]None[illegible]	click_and_hold(gmail_link)
double_click(on_element=None)	[illegible]	on_element: [illegible]None[illegible]	double_click(info_box)
drag_and_drop(source, target)	[illegible]	source: [illegible] target: [illegible]	drag_and_drop(img, canvas)
key_down(value, element=None)	[illegible]Ctrl[illegible]Alt[illegible]Shift[illegible]	key:[illegible]Key[illegible]Keys[illegible] target[illegible]None[illegible]	key_down(Keys. SHIFT)\ send_keys(‘n’)\ key_up(Keys.SHIFT)
key_up(value, element=None)	[illegible]	key:[illegible]Key[illegible]Keys[illegible]	

		□□ target□□□□□□□□□□□□□□□□□□□□None□□□□□□□□□□□□□□□□□□□□	
move_to_element(to_element)	□□□□□□□□□□□□□□□□	to_element: □□□□□□	move_to_element(gmail_link)
perform()	□□□□□□□□□□□□□□□□		perform()
release(on_element=None)	□□□□□	on_element: □□□□□□□□□□	release(banner_img)
send_keys(keys_to_send)	□□□□□□□□□□□□□□□□	keys_to_send: □□□□□□□	send_keys(“hello”)
send_keys_to_element(element, keys_to_send)	□□□□□□□□□□□□□□□□□□	element: □□□□□□□ keys_to_send: □□□□□□□	send_keys_to_element(firstName, “John”)

□□□□□□□□□□

http://selenium.googlecode.com/git/docs/api/py/webdriver/selenium. webdriver.common.action_chains.html□

Interactions API[illegible]Safari[illegible]https://code.google.com/p/selenium/wiki/AdvancedUserInteractions[illegible]

9.1.1 [illegible]

[illegible]Shift+N[illegible]label[illegible]

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.keys import Keys
import unittest

class HotkeyTest(unittest.TestCase):
   URL = "https://rawgit.com/jeresig/jquery.hotkeys/master/test-static-
05.html"

   def setUp(self):
      self.driver = webdriver.Chrome()
      self.driver.get(self.URL)
      self.driver.implicitly_wait(30)
      self.driver.maximize_window()

   def test_hotkey(self):
      driver = self.driver

      shift_n_label = WebDriverWait(self.driver, 10).\
         until(expected_conditions.visibility_of_element_
         located((By.ID, "_shift_n")))

      ActionChains(driver).\
         key_down(Keys.SHIFT).\
         send_keys('n').\
         key_up(Keys.SHIFT).perform()
      self.assertEqual("rgba(12, 162, 255, 1)",
                       shift_n_label.value_of_css_
                       property("background-color"))

   def tearDown(self):
      self.driver.close()
```

```
if  __name__ == "__main__":
    unittest.main(verbosity=2)
```

□□□□ActionChains□□□□□□□□□□□□□□□□□□□□□□□□□□□key_down()□send_key()□key_up()□□□□□□□□□□Shift+N□□□□

```
ActionChains(driver).\
   key_down(Keys.SHIFT).\
   send_keys('n').\
   key_up(Keys.SHIFT).perform()
```

□□□ActionChains□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□perform()□□□□□□□□□□□□□□□□

9.1.2 □□□□

□□□□□□□□ActionChains□□□move_to_element()□□□□□□□□□□□□□□□□□□□onMouseOver□□□move_to_element()□□□□□□□□□□□□□□□□□□□□

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.common.action_chains import ActionChains
import unittest


class ToolTipTest (unittest.TestCase):
   def setUp(self):
      self.driver = webdriver.Firefox()
      self.driver.get("http://jqueryui.com/tooltip/")
      self.driver.implicitly_wait(30)
      self.driver.maximize_window()

   def test_tool_tip(self):
      driver = self.driver

      frame_elm = driver.find_element_by_class_name("demo-frame")
      driver.switch_to.frame(frame_elm)

      age_field = driver.find_element_by_id("age")
      ActionChains(self.driver).move_to_element(age_field).perform()
```

```
        tool_tip_elm = WebDriverWait(self.driver, 10)\
            .until(expected_conditions.visibility_of_element_
            located((By.CLASS_NAME, "ui-tooltip-content")))

        # verify tooltip message
        self.assertEqual("We ask for your age only for statistical
        purposes.", tool_tip_elm.text)

    def tearDown(self):
        self.driver.close()

if __name__ == "__main__":
    unittest.main(verbosity=2)
```

9.1.2.1 □□□□

□□ActionChains□□□double_click()□□□□□□□□□□□□□□□□□□□□□□

```
from selenium import webdriver

from selenium.webdriver.common.action_chains import ActionChains
import unittest

class DoubleClickTest (unittest.TestCase):
    URL = "http://api.jquery.com/dblclick/"

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.get(self.URL)
        self.driver.maximize_window()

    def test_double_click(self):
        driver = self.driver
        frame = driver.find_element_by_tag_name("iframe")
        driver.switch_to.frame(frame)
        box = driver.find_element_by_tag_name("div")

        # verify color is Blue
        self.assertEqual("rgba(0, 0, 255, 1)",
                         box.value_of_css_property("background-color"))

        ActionChains(driver).move_to_element
           ( driver.find_element_by_tag_name("span"))\.perform()

        ActionChains(driver).double_click(box).perform()

        # verify Color is Yellow
        self.assertEqual("rgba(255, 255, 0, 1)",
                        box.value_of_css_property("background-color"))

    def tearDown(self):
        self.driver.close()
```

```
if __name__ == "__main__":
    unittest.main(verbosity=2)
```

9.1.2.2 [illegible]

[illegible]ActionChains[illegible]drag_and_drop()[illegible]

[illegible]

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
import unittest


class DragAndDropTest (unittest.TestCase):

    URL = "http://jqueryui.com/resources/
          demos/droppable/default.html"

    def setUp(self):
        self.driver = webdriver.Firefox()
        self.driver.get(self.URL)
        self.driver.maximize_window(30)
        self.driver.maximize_window()

    def test_drag_and_drop(self):
        driver = self.driver

        source = driver.find_element_by_id("draggable")
        target = driver.find_element_by_id("droppable")

        ActionChains(self.driver).drag_and_drop(source, target). perform()
        self.assertEqual("Dropped!", target.text)

    def tearDown(self):
        self.driver.close()

if __name__ == "__main__":
    unittest.main(verbosity=2)
```

9.2 [illegible]JavaScript

[illegible]JavaScript[illegible]WebDriver[illegible]JavaScript[illegible]

[illegible]WebDriver[illegible]

□ □	□ □	□ □	□ □
execute_async_script(script, *args)	□□□□ JS□□	script:□□ □□JS□□□ args:JS□ □□□□□□□	driver.execute_async_script("return document.title")
execute_ script(script, *args)	□□□□ JS□□	script:□□ □□JS□□□ args:JS□ □□□□□□□	driver.execute_ script("return document.title")

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□JavaScript□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
from selenium import webdriver
import unittest


class ExecuteJavaScriptTest (unittest.TestCase):
   def setUp(self):
      # create a new Firefox session
      self.driver = webdriver.Firefox()
      self.driver.implicitly_wait(30)
      self.driver.maximize_window()

      # navigate to the application home page
      self.driver.get("http://demo.magentocommerce.com/")

   def test_search_by_category(self):

      # get the search textbox
      search_field = self.driver.find_element_by_name("q")
      self.highlightElement(search_field)
      search_field.clear()

      # enter search keyword and submit
      self.highlightElement(search_field)
      search_field.send_keys("phones")
      search_field.submit()

      # get all the anchor elements which have product names
      # displayed currently on result page using
```

```
        # find_elements_by_xpath method
        products = self.driver.find_elements_by_xpath("//h2[@
class='product-name']/a")

        # check count of products shown in results
        self.assertEqual(2, len(products))

    def tearDown(self):
        # close the browser window
        self.driver.quit()

    def highlightElement(self, element):
        self.driver.execute_script("arguments[0].setAttribute('style',
        arguments[1]);",
        element, "color: green;
        border: 2px solid green;")
        self.driver.execute_script("arguments[0].setAttribute('style',
        arguments[1]);",
        element , "")

if __name__ == "__main__":
    unittest.main(verbosity=2)
```

□□□□□□□□WebDriver□□execute_script□□□□□JavaScript□□□□□□□□□□□□□□□□□JavaScript□□□

```
def highlightElement(self, element):
     self.driver.execute_script("arguments[0].setAttribute('style',
     arguments[1]);",
     element, "color: green; border: 2px solid green;")
     self.driver.execute_script("arguments[0].setAttribute('style',
     arguments[1]);",
     element , "")
```

9.3 □□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□WebDriver□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□ □	□ □	□ □	□ □

[illegible]	[illegible]	[illegible]	[illegible]
save_ screenshot(filename)	[illegible]	filename[illegible]/[illegible]	Driver.save_ screenshot ("homepage.png")
get_screenshot_as_base64()	[illegible] base64 [illegible] HTML[illegible] base64 [illegible]		driver.get_screenshot_as_base64()
get_screenshot_as_file(filename)	[illegible] IOError[illegible] False[illegible] True	filename[illegible]/[illegible]	driver.get_ screenshot_as_file('/results/ screenshots/ HomePage.png')
get_screenshot_as_png()	[illegible]		driver.get_screenshot_as_png()

[illegible]NoSuchElement Exception[illegible]bug[illegible]

```
from selenium import webdriver
import datetime, time, unittest
from selenium.common.exceptions import NoSuchElementException


class ScreenShotTest(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Firefox()
        self.driver.get("http://demo.magentocommerce.com/")

    def test_screen_shot(self):
        driver = self.driver
        try:
            promo_banner_elem = driver.find_element_by_id("promo_ banner")
            self.assertEqual("Promotions", promo_banner_elem.text)
        except NoSuchElementException:
            st = datetime.datetime\
                .fromtimestamp(time.time()).strftime('%Y%m%d_%H%M%S')
            file_name = "main_page_missing_banner" + st + ".png"
            driver.save_screenshot(file_name)
            raise

    def tearDown(self):
        self.driver.close()

if __name__ == "__main__":
    unittest.main(verbosity=2)
```

□□□□□□□□□□□□□□□□“promo_banner”□□□□□□□□□□save_screenshot()□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
try:
    promo_banner_elem = driver.find_element_by_id("promo_banner")
    self.assertEqual("Promotions", promo_banner_elem.text)
except NoSuchElementException:
    st = datetime.datetime.fromtimestamp(time.time()).
strftime('%Y%m%d_%H%M%S')
    file_name = "main_page_missing_banner" + st + ".png"
    driver.save_screenshot(file__name)
    raise
```

> □□PNG□□□□□□□□□□□□□□□□□□□□

9.4 □□□□

□□

□□□Selenium WebDriver□□□□□□□□□□Python□□□□Castro□□□□□Selenium□□Jason Huggin□□□Castro□□□□□□□□□Pyvnc2swf□□□□□VNC□□□□□□SWF□□□

□□□VNC□□□□□□□□□□□□□□VNC□□□□□□□□□□PyGame□□□□Castro□pip□□□□

```
pip install Castro
```

□□Server□Viewer□□□Windows□□□□□□□□TightVNC□□□

□□□Ubuntu□□□□□□□□**Settings | Preference | Remote Desktop**□□□□**Allow other users to view your desktop**□□□□Mac□□□□□□Vine VNC Server□□□**System Preferences**□□□**Remote Desktop**□

□□□□□□□□□□□□□□□□□□□□□

```
import unittest
from selenium import webdriver
from castro import Castro

class SearchProductTest(unittest.TestCase):
    def setUp(self):
        # create an instance of Castro and provide name for the output
        # file
        self.screenCapture = Castro(filename="testSearchByCategory. swf")
        # start the recording of movie
        self.screenCapture.start()

        # create a new Firefox session
        self.driver = webdriver.Firefox()
```

```
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

        # navigate to the application home page
        self.driver.get("http://demo.magentocommerce.com/")

    def test_search_by_category(self):

        # get the search textbox
        search_field = self.driver.find_element_by_name("q")
        search_field.clear()

        # enter search keyword and submit
        search_field.send_keys("phones")
        search_field.submit()

        # get all the anchor elements which have product names
        # displayed
        # currently on result page using find_elements_by_xpath method
        products = self.driver.find_elements_by_xpath("//h2[@
        class='product-name']/a")

        # check count of products shown in results
        self.assertEqual(2, len(products))

    def tearDown(self):
        # close the browser window
        self.driver.quit()
        # Stop the recording
        self.screenCapture.stop()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

[illegible]Castro[illegible]start()[illegible]stop()[illegible]setUp()[illegible]Castro[illegible]

```
def setUp(self):
    # Create an instance of Castro and provide name for the output
    # file
    self.screenCapture = Castro(filename="testSearchByCategory.swf")
    # Start the recording of movie
    self.screenCapture.start()

    # create a new Firefox session
    self.driver = webdriver.Firefox()
    self.driver.implicitly_wait(30)
    self.driver.maximize_window()

    # navigate to the application home page
    self.driver.get("http://demo.magentocommerce.com/")
```

[illegible]teadDown()[illegible]stop()[illegible]

```
def tearDown(self):
    # close the browser window
    self.driver.quit()
    # Stop the recording
    self.screenCapture.stop()
```

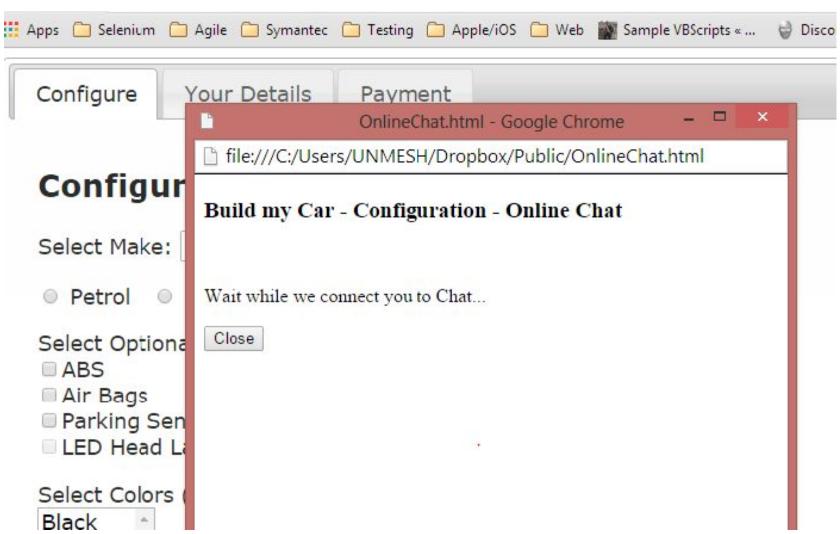
[illegible]setUp() [illegible] tearDown()[illegible]

9.5 [illegible]

[illegible]Driver Context[illegible]

[illegible]WebDriver Context[illegible]

[illegible]



[illegible]PopupWindowTest[illegible]test_popup_window()[illegible]

[illegible]

```
from selenium import webdriver
import unittest

class PopupWindowTest(unittest.TestCase):

    URL =
"https://rawgit.com/upgundecha/learnsewithpython/master/pages/Config.html"

    def setUp(self)     :
        self.driver = webdriver.Firefox()
        self.driver.get(self.URL)
        self.driver.maximize_window()

    def test_popup_ window(self):
        driver = self.driver

        # save the WindowHandle of Parent Browser Window
        parent_window_id = driver.current_window_handle

        # clicking Help Button will open Help Page in a new Popup
        # Browser Window
        help_button = driver.find_element_by_id("helpbutton")
        help_button.click()
        driver.switch_to.window("HelpWindow")
        driver.close()
        driver.switch_to.window(parent_window_id)

    def tearDown(self):
        self.driver.close()

if __name__ == "__main__":
    unittest.main(verbosity=2)
```

[illegible]Context[illegible]current_window_handle[illegible]WebDriver[illegible]switch_to.window()[illegible]

```
driver.switch_to_window("HelpWindow")
```

[illegible]Help[illegible]close()[illegible]

```
driver.close()

# switch back to Home page window using the handle
driver.switch_to_window(default_window)
```

9.6 □□cookies

□□□□□□□□□□cookies□□Web□□□WebDriver□□□□□□□cookies□□□□□□□□□□□□□□cookies□□□□□□□□□□□□□□□cookies□□□□Web□□□□□□□□□□□□□□□□□□

□ □	□ □	□ □	□ □
add_ cookie(cookie_dict)	□□□□□□□□ cookie□□	cookie_ dict:□ □□□□□□name□ value□	driver.add_ cookie({"foo","bar"})
delete_all_cookies()	□□□□□□□□□□ cookie□□		driver.delete_all_cookies()
delete_cookie(name)	□□□□□□name□ cookie□□	name:□□□□ cookie□□□	driver.delete_cookie("foo")
get_cookie(name)	□□□□□□name□ cookie□□□□□□□ □□□□□none	name:□□□□ cookie□□□	driver.get_ cookie("foo")
get_cookies()	□□□□□□□□□ cookie□□		driver.get_cookies()

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□cookie□□

```
import unittest
from selenium import webdriver
from selenium.webdriver.support.ui import Select


class CookiesTest(unittest.TestCase):
    def setUp(self):
        # create a new Firefox session
        self.driver = webdriver.Firefox()
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

        # navigate to the application home page
        self.driver.get("http://demo.magentocommerce.com/")

    def test_store_cookie(self):
        driver = self.driver
        # get the Your language dropdown as instance of Select class
        select_language = \
            Select(self.driver.find_element_by_id("select-language"))

        # check default selected option is English
        self.assertEqual("ENGLISH", select_language.first_selected_
option.text)
        # store cookies should be none
        store_cookie = driver.get_cookie("store")
        self.assertEqual(None, store_cookie)

        # select an option using select_by_visible text
        select_language.select_by_visible_text("French")

        # store cookie should be populated with selected country
        store_cookie = driver.get_cookie("store")['value']
        self.assertEqual("french", store_cookie)

    def tearDown(self):
        # close the browser window
        self.driver.quit()

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

□□□□□□□□□□□□cookie□□□□□□□□□get_cookie()□□□□□□□cookie□□□

9.7 □□□□

□□□□□□□□□□□□□Selenium WebDriver□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□cookies□

[illegible]ActionChains[illegible]

[illegible]JavaScript[illegible]Ajax[illegible]JavaScipt API[illegible]

[illegible]bug[illegible]bug[illegible]

[illegible]cookies[illegible]

[illegible]

[illegible]10[illegible] [illegible]

Selenium WebDriver Python API[illegible]Selenium WebDriver[illegible]unittest[illegible]unittest[illegible]Selenium WebDriver[illegible]Selenium WebDriver [illegible]

[illegible]Selenium WebDriver[illegible]BDD[illegible]BDD[illegible]

[illegible]Selenium Python API[illegible]CI[illegible]

[illegible]

- □□□□□Behave□
- □□Behave□□feature□
- □□Behave□Selenium WebDriver□□□□□feature□
- □□□□□Jenkins□
- □□Jenkins□□Selenium□□□
- □□Jenkins□□□□□□□

10.1 □□□□□□□BDD□

BDD□Dan North□□□□□Introducing BDD□□□□□□□□□□□□□□

BDD□□□□□□□□□□ATDD□□□□□□□□□□□story testing□□□□□□□specification by example□□BDD□□□□□□□□□□QA □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Python□□□□□□□□□□□□BDD□□□□□□□□□□□□Behave□Lettuce□

Lettuce□□□□□□□Ruby BDD Cucumber□□□

□□□□□□□□□□□□□□□□□□□□□Behave□□□□□□□□□□BDD□

10.1.1 Behave□□

□□Behave□□□□□□□□□□□□□□□□□□□□□□□□□□□□Behave□

```
pip install behave
```

□□□□□□□□□□□□□□□□□□□Behave□□□□□□□□□□

10.1.2 □□□feature

□Behave□□□□□□feature□□□□□□□□□□□□□□□□□□feature□User Story□□feature□User Story□□□□□□□□□Behave□□□Given-When-Then□GWT□□□□Gherkin□□□□feature□□□

□□□□□□□□□□□□□□□□□feature□□□□□□feature□□□□□□□□□□□□ feature□□□GWT□□□User Story□□□□□□□□□□□□□□□□□□□□□Scenario Outline□□□□□□□□□□Scenario Steps□□□□□□□

- **Given**□□□□□□□□□□□□□□□□□□——□□□□□□
- **When**□□□□□□□□□□□□□□□□□□——□□□□□□□
- **Then**□□□□□□□□□□□□□□□□□——□□□□□□□□□□□□□□□□□□

□□□□□□□□□□When□Then□

```
Feature: I want to search for products

 Scenario Outline: Search
   Given I am on home page
    when I search for "phone"
    then I should see list of matching products in
search results
```

□□□□□□□□□□□□□“.feature”□□□□□□□□□ Behave□□□□□□□□□bdd/feature□□□□□□□ feature□□□□□“search.feature”□□□□□□□□□□

10.1.2.1 step□□

feature□□□□□□□□□□feature□□□□□□□□□ step□□□□□□□□step□□□□□□Python□□□□□□□□ □□□□□□□□□□□□□□□Python API□□Selenium WebDriver□□□□□□step□□□□step□□□□□□□□ feature□□□□□□□□□□“steps”□□□□□□□□□ search_steps.py□□□□□□□feature□□□□step□

```
from behave import *

@given('I am on home page')
def step_i_am_on_home_page(context):
```

```
context.driver.get("http://demo.magentocommerce.com/")

@when('I search for {text}')
def step_i_search_for(context, text):
    search_field = context.driver.find_element_by_name("q")
    search_field.clear()

    # enter search keyword and submit
    search_field.send_keys(text)
    search_field.submit()

@then('I should see list of matching products in search results')
def step_i_should_see_list(context):
    products = context.driver.\

find_elements_by_xpath("//h2[@class='product-name']/a")
    # check count of products shown in results
    assert len(products) > 0
```

□□□□GWT□□□□□□□□□□□□□□step□□□□□□□□□□Given□□“I am on home page”□□□step□□□□□@□□□□feature□□□□□□GWT□□□□□□step□□@given□@when□@then□□□□□□□□□□□□□□□□□□□□step□□□□□□□□□□□□□□□□□□□□□□□□“I am on home page”□

```
@given('I am on home page')
def step_i_am_on_home_page(context):

context.driver.get("http://demo.magentocommerce.com/")
```

□□□□□□step□□□□□□□□□step□□□□□□□@when□□□□□□□□□□□□□□when I search for“phone”□□□□step□□□□□{text}□□□□□“phone”□□□□□□□□□

```
@when('I search for {text}')
def step_i_search_for(context, text):
    search_field =
context.driver.find_element_by_name("q")
    search_field.clear()

    # enter search keyword and submit
    search_field.send_keys(text)
    search_field.submit()
```

□□□□□□□□□□□□□□step□□□□□□□□□Behave□□□□□□□□□□□□□□□□□□□□□□□3□□□□□Behave□□□□□□□□□□□□□□□□□□□□□□□step□□□□□□

10.1.2.2 □□□□

□□□feature□□□□□□□□□□□□□□□□□□□□

Behave□□□□□□□□step□□□step□□□□□□□□□□
□□□□□Selenium WebDriver□Firefox□□□□□□□
□□□□□□□□□□□□□□Firefox□□□□WebDriver□□□□
□□feature□□□□□□□□□□□□□□□
environment.py□□□before_all()□after_all()□
□□□□□□□□□□feature□□□□□□□□□□

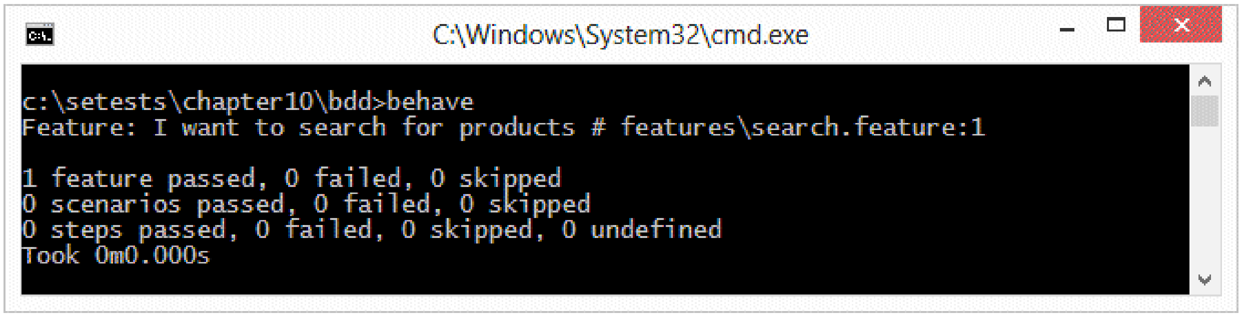
```
from selenium import webdriver

def before_all(context):
    context.driver = webdriver.Chrome()

def after_all(context):
    context.driver.quit()
```

10.1.2.3 □□feature

□□□□□□□□Behave□□feature□□□□□□□□□
□□□□□□□□□□□□□□□□bdd□□□□□□□□□□
□“behave”□□□

```
behave
```

Behave[illegible]bdd[illegible]feature[illegible]step[illegible]scenario[illegible]behave[illegible]



Behave[illegible]3[illegible]feature[illegible]scenario[illegible]step[illegible]

10.1.2.4 [illegible]

[illegible]

[illegible]search.feature[illegible]“Example”[illegible]

□1□□□□□□□□□□□□□□□□□□□□□□□“□□”□□□□□“□□□□□”□□□“Example”□□□□□□□□□□□□□□□□□□□□

```
Feature: I want to search for products

  Scenario Outline: Search
    Given I am on home page
     when I search for <term>
     then I should see results <search_count> in
search results

Examples: By category
  |term    | search_count |
  |Phones   |  2           |
  |Bags    |  7           |

Examples: By product name
  | term          | search_count |
  | Madison earbuds |   3        |
```

□2□□□search_steps.py□□□□□□□□step□

```
from behave import *

@given('I am on home page')
def step_i_am_on_home_page(context):

context.driver.get("http://demo.magentocommerce.co
m/")

@when('I search for {text}')
def step_i_search_for(context, text):
   search_field =
```

```
context.driver.find_element_by_name("q")
    search_field.clear()

    # enter search keyword and submit
    search_field.send_keys(text)
    search_field.submit()

@then('I should see results {text} in search
results')
def step_i_should_see_results(context, text):
    products = context.driver.\

find_elements_by_xpath("//h2[@class='product-
name']/a")
    # check count of products shown in results
    assert len(products) >= int(text)
```

□□□feature□□Behave□□search.feature□□□“Example”□□□□□□□□□□□□□□□□□□□□□□“Example”□□□□□□□scenario□□step□□□□□□step□□□□□□□□□□Behave□□□□□□feature□□□□□□□□□□feature□□□□□□□□□□

Behave□□□□□□□□ -junit□□junit□□□□□

```
C:\Users\amitr\Desktop\setests_final\bdd>behave
Feature: I want to search for products # features\search.feature:1

  Scenario Outline: Search                          # features\search.feature:3
    Given I am on home page                         # steps\search_steps.py:4
    When I search for Phones                        # steps\search_steps.py:9
    Then I should see results 2 in search results # steps\search_steps.py:18

  Scenario Outline: Search                          # features\search.feature:3
    Given I am on home page                         # steps\search_steps.py:4
    When I search for Bags                          # steps\search_steps.py:9
    Then I should see results 7 in search results # steps\search_steps.py:18

  Scenario Outline: Search                          # features\search.feature:3
    Given I am on home page                         # steps\search_steps.py:4
    When I search for Madison earbuds               # steps\search_steps.py:9
    Then I should see results 3 in search results # steps\search_steps.py:18

1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
9 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m14.539s

C:\Users\amitr\Desktop\setests_final\bdd>
```

10.2 □□□□Jenkins

Jenkins□Java□□□□□□□□□□□CI□□□□□□□□ Hudson□□□Jenkins□Hudson□□□□□

Jenkins□□□□□□□□□□□□□□□CVS□SVN□Git□ Mercurial□Perforce□ClearCase□□□□□□□□

Apache Ant□Java Maven□□□□□□□□□□□□□□□□□
□□□□shell□□□Windows□□□□□□□□□□□□□□□□□

Jenkins□□□□□□□□□□□□□□□□□□□□□□□□□□□□
Selenium WebDriver□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□Jenkins□□□□□□□□□□
□□□□□□□□□□

10.2.1 Jenkins□□□□

□□□□□□□□Jenkins□□□□□□□□□□□□□□□□□□□□
□□□Jenkins□□□□□□□□□□□□□□□□□□□□□□□□
Jenkins Dashboard□□□□□□□□□□□□□□□□□□□2□
□□□□□□□□□

□□□□□unittest□TestSuite Runner□□□□□
□□□□JUnit□□□□□□□□□□□□□□□□xmlrunner□
Python□□□□□□□□

https://pypi.python.org/pypi/xmlrunner/1.7.4□□□

□□□□□□□□□□□□□□xmlrunner□

```
pip install xmlrunner
```

□□□□□□smoketests.py□□□TestSuite Runner□□homepagetests.py□searchtest.py □□□□□□□□□□□□□□□□xmlrunner.XMLTestRunner□□□□□□□□□□JUnit□□□□□□□□□XML □□□□□□□□test-reports□□□□□□□□□ xmlrunner□□□□smoketest.py□□□□□□□□□□□□□□□□□□□

```
import unittest
from xmlrunner import xmlrunner
from searchtest import SearchProductTest
from homepagetests import HomePageTest

# get all tests from SearchProductTest and
HomePageTest class
search_tests =
unittest.TestLoader().loadTestsFromTestCase(Search
Prod uctTest)
```

```
home_page_tests =
unittest.TestLoader().loadTestsFromTestCase(HomePa
g eTest)

# create a test suite combining search_test and
home_page_test
smoke_tests = unittest.TestSuite([home_page_tests,
search_tests])

# run the suite
xmlrunner.XMLTestRunner(verbosity=2, output='test-
reports').run(smoke_ tests)
```

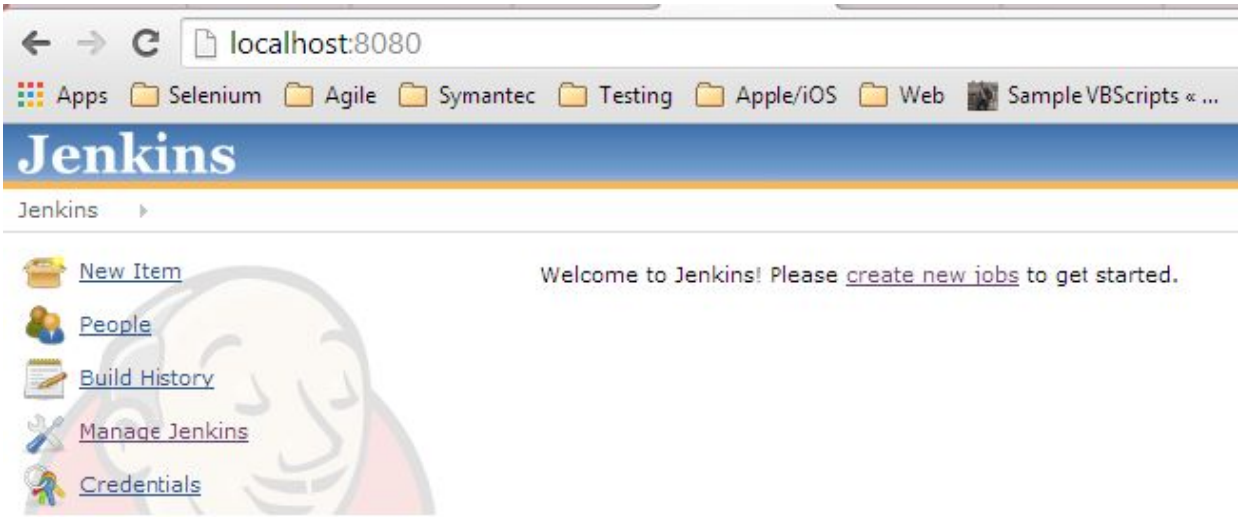
10.2.2 □□Jenkins

□□Jenkins□□□□□□□□□□□□□□Jenkins□□□□□□□□□□□□□□□□□□□□□□Jenkins□□□□□□□□□□□□□□□□□□□□□□□□□□

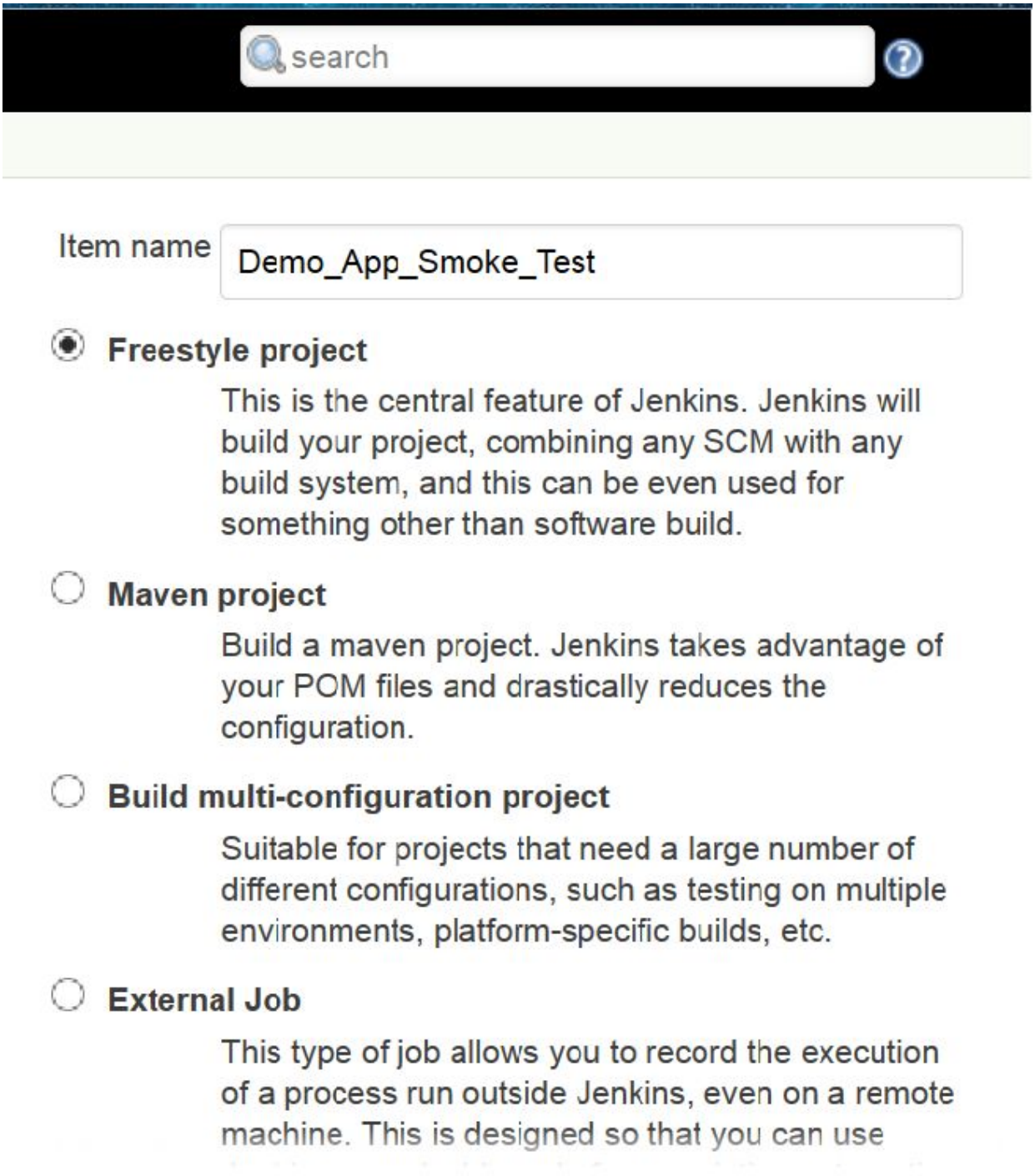
□1□□□□□□Jenkins CI□□□□□□□□□Jenkins Windows□□□□□□Windows 7□□□Jenkins□

□2□□□□□□□**Jenkins Dashboard**□□□□□□□□□http:// localhost:8080□□

□3□□**Jenkins Dashboard**□□□□□□□□□□
□**New Item**□□□□□□□□**create new jobs**□□
□□□□□□□□Jenkins□□□□□□□□□



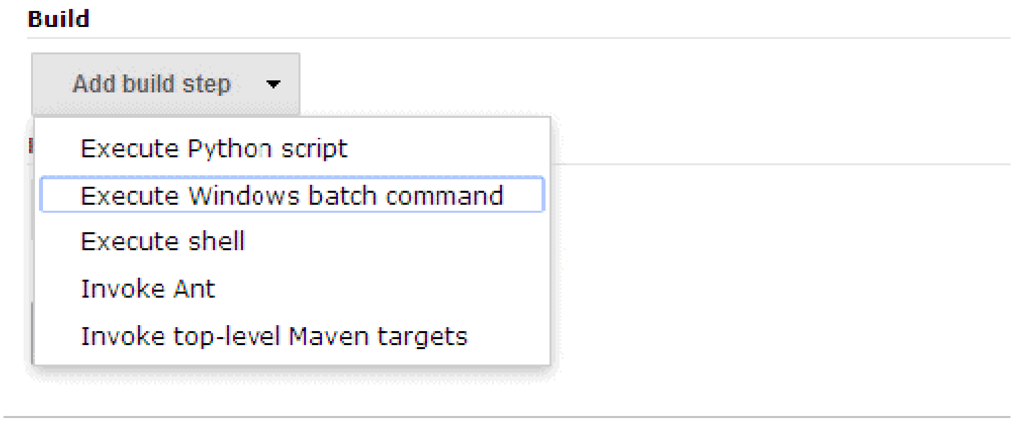
□4□□□□□□**Item name**□□□□□□□
Demo_App_Smoke_Test□□□□□□□□□□□□□□□□□
□**Freestyle project**□□□□□□□□□□□


search
Item name
Demo_App_Smoke_Test
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
Build multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use

□5□□□□□□OK□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

> [illegible]SCM[illegible]SVN[illegible]Git[illegible]Perforce[illegible]Jenkins[illegible]Windows[illegible]Execute Windows batch command[illegible]Jenkins[illegible]

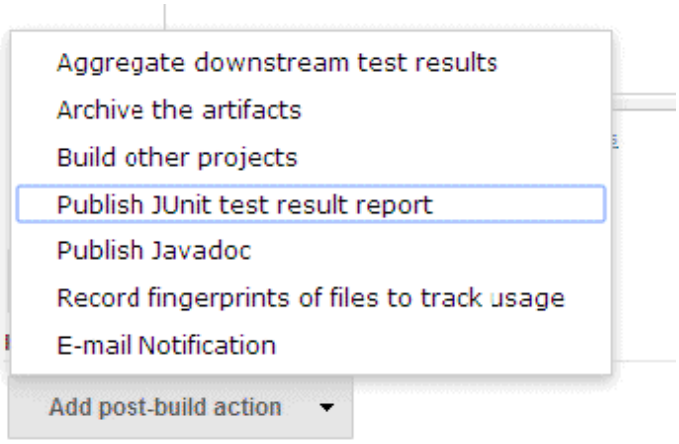
[illegible]6[illegible]Bulid[illegible]Add build step[illegible]Windows[illegible]Execute Windows batch command[illegible]



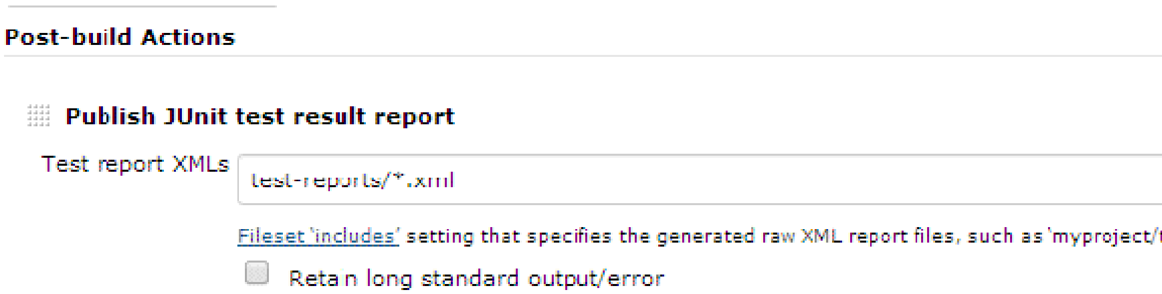
[illegible]7[illegible]Command[illegible]
[illegible]Python[illegible]
[illegible]Jenkins[illegible]smoketest.py[illegible]

```
copy c:\setests\chapter10\smoketests\*.py
python smoketest.py
```

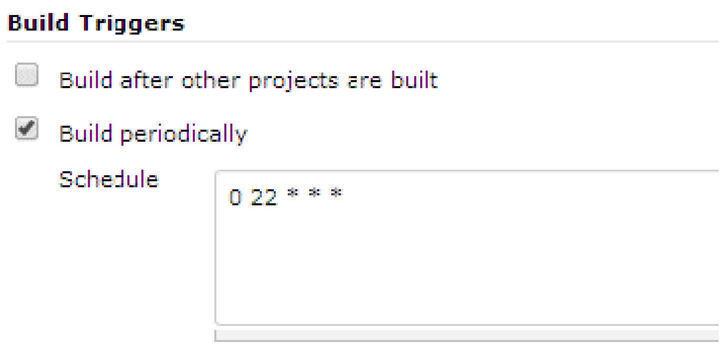
□8□□□□□□□□□□□smoketest.py□□□JUnit□□□□□□□□□□□□□□□□□□□□Jenkins Dashboard□□□□□Jenkins□□□□□□□□□□□□□□□□□□□□□□Add post-build action□□□□□□□□JUnit□□□□□□□Publish JUnit test result report□□□□□□□□□□



□9□□□□□□□□Post-build Actions□□□□□□□□□□XML□Test report XMLs□□□□□□□test-reports/ *.xml□□□□□□□Jenkins□□□□□□□□□□□□□□test-reports□□□□□□□□□□□□



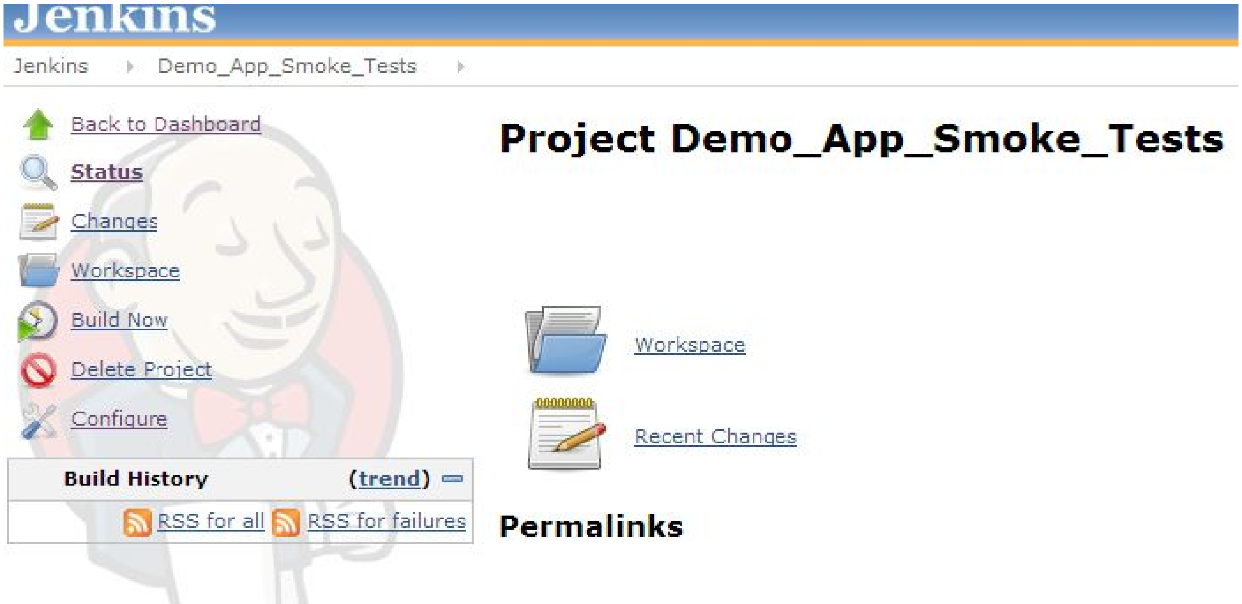
□10□□□□□□□□□□□□□□□□□□□□□□Build Triggers□□□□□□□□□□Build periodically□□□□□□□Schedule□□□□□□□□□□□□□□□□□□□□22□□□□□□□□□□□□□□□□□□□□□□□□□□Jenkins□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□



□11□□□□□□□□□□□□□□Jenkins□□□□□□□□□□□□□□□

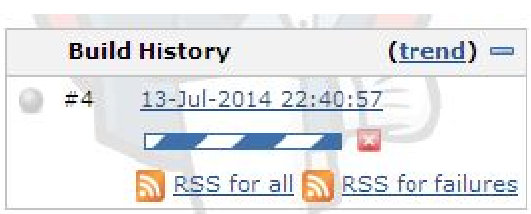
[illegible]12[illegible]

[illegible]Build Now[illegible]



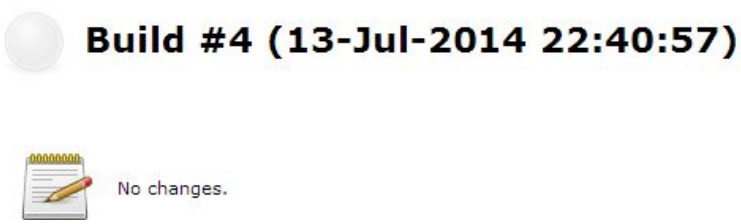
[illegible]13[illegible]Build History[illegible]

[illegible]



[illegible]14[illegible]Build History[illegible]

[illegible]



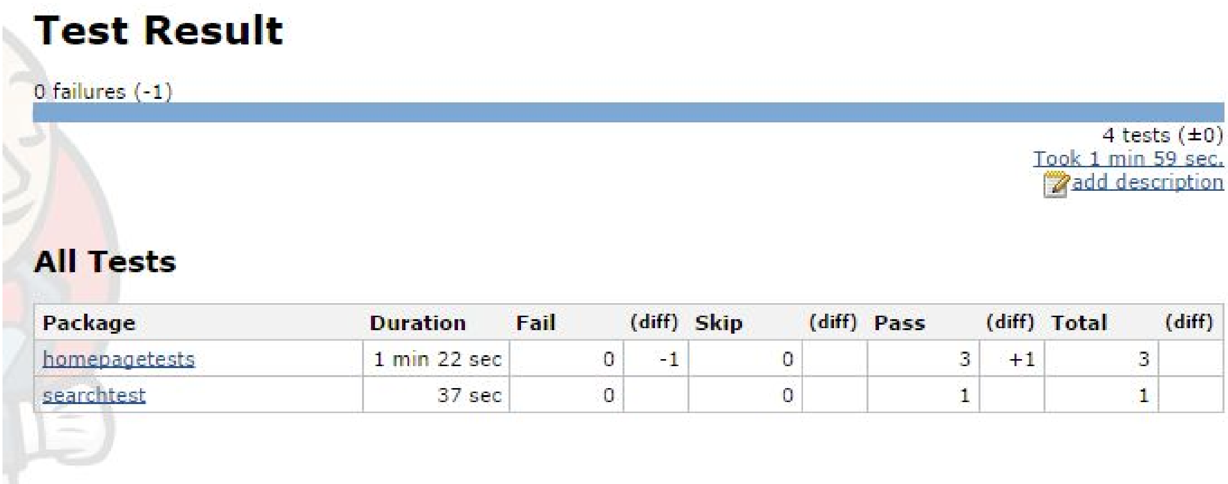
□15□□□Jenkins□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Console Output□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□“□□□□□”□□□

```
Started by user anonymous
Building in workspace C:\Program Files (x86)\Jenkins\workspace\Demo_App_Smoke_Test_1
[Demo_App_Smoke_Test_1] $ cmd /c call C:\Windows\TEMP\hudson4452624957017176027.bat

C:\Program Files (x86)\Jenkins\workspace\Demo_App_Smoke_Test_1>copy C:\Users\amitr\Desktop\Mrunmayee\Final
\setests_final\smoketests\*py
C:\Users\amitr\Desktop\Mrunmayee\Final\setests_final\smoketests\homepagetests.py
C:\Users\amitr\Desktop\Mrunmayee\Final\setests_final\smoketests\searchtests.py
C:\Users\amitr\Desktop\Mrunmayee\Final\setests_final\smoketests\smoketests.py
        3 file(s) copied.

C:\Program Files (x86)\Jenkins\workspace\Demo_App_Smoke_Test_1>python smoketests.py
Build was aborted
Aborted by anonymous
Recording test results
ERROR: Publisher hudson.tasks.junit.JUnitResultArchiver aborted due to exception
hudson.AbortException: No test report files were found. Configuration error?
        at hudson.tasks.junit.JUnitParser$ParseResultCallable.invoke(JUnitParser.java:116)
        at hudson.tasks.junit.JUnitParser$ParseResultCallable.invoke(JUnitParser.java:92)
        at hudson.FilePath.act(FilePath.java:981)
        at hudson.FilePath.act(FilePath.java:959)
        at hudson.tasks.junit.JUnitParser.parseResult(JUnitParser.java:89)
        at hudson.tasks.junit.JUnitResultArchiver.parse(JUnitResultArchiver.java:121)
        at hudson.tasks.junit.JUnitResultArchiver.perform(JUnitResultArchiver.java:138)
        at hudson.tasks.BuildStepCompatibilityLayer.perform(BuildStepCompatibilityLayer.java:74)
        at hudson.tasks.BuildStepMonitor$1.perform(BuildStepMonitor.java:20)
        at hudson.model.AbstractBuild$AbstractBuildExecution.perform(AbstractBuild.java:770)
        at hudson.model.AbstractBuild$AbstractBuildExecution.performAllBuildSteps(AbstractBuild.java:734)
        at hudson.model.Build$BuildExecution.post2(Build.java:183)
        at hudson.model.AbstractBuild$AbstractBuildExecution.post(AbstractBuild.java:683)
        at hudson.model.Run.execute(Run.java:1784)
        at hudson.model.FreeStyleBuild.run(FreeStyleBuild.java:43)
        at hudson.model.ResourceController.execute(ResourceController.java:89)
        at hudson.model.Executor.run(Executor.java:240)
Finished: ABORTED
```

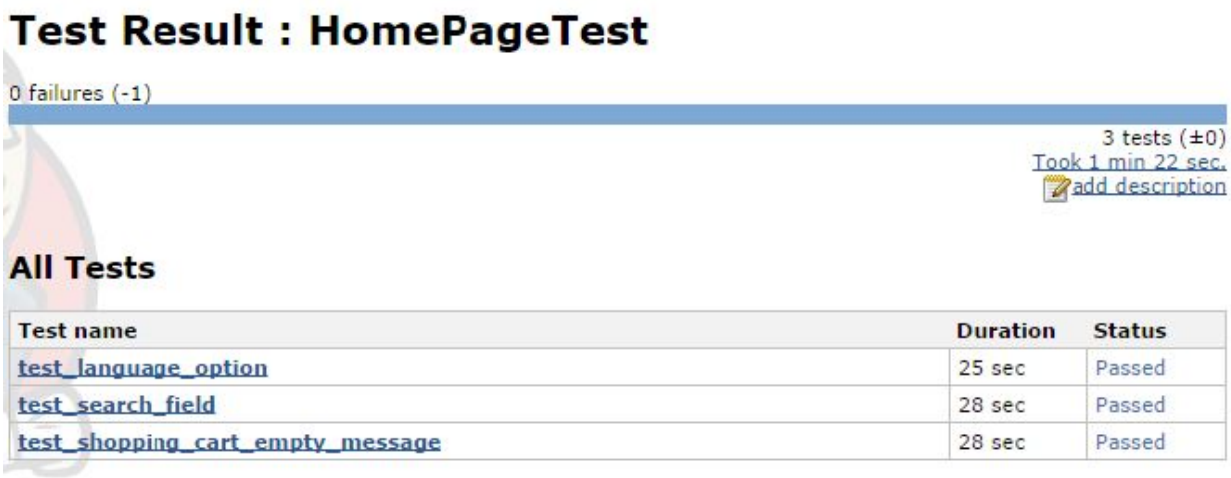
□16□□□Jenkins□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□

□17□Jenkins□□□□unittest□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Test
Results□□□□□□□Jenkins□□□□□□□□

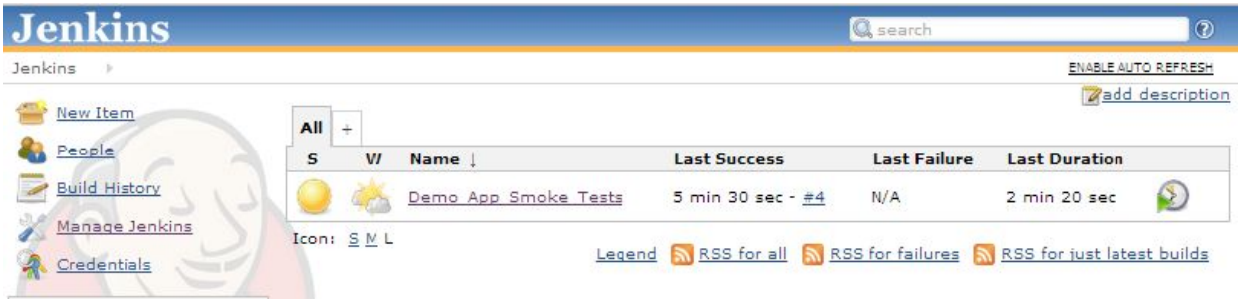
□18□□□□□□□□□□□□□JUnit□□□□□□□□□□□□
□Test Results□□□Jenkins□□□□JUnit□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□



□19□□□□□□□□Package□□□□□□□□□□□□□□
□□□□□□□□□□□□



Jenkins□□□□□□□□□□Dashboard□□□□□□□□□□□□□□□□□□□



10.3 □□□□

□□□□□□□□□□□□□□Selenium□BDD□Behave□□CI□Jenkins□□□□□□□□Selenium

WebDriver API□Behave□□□□□□□□feature□
step□□□□□□□□□□□□□□□□

□□□□Jenkins□□Selenium WebDriver□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□Jenkins□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

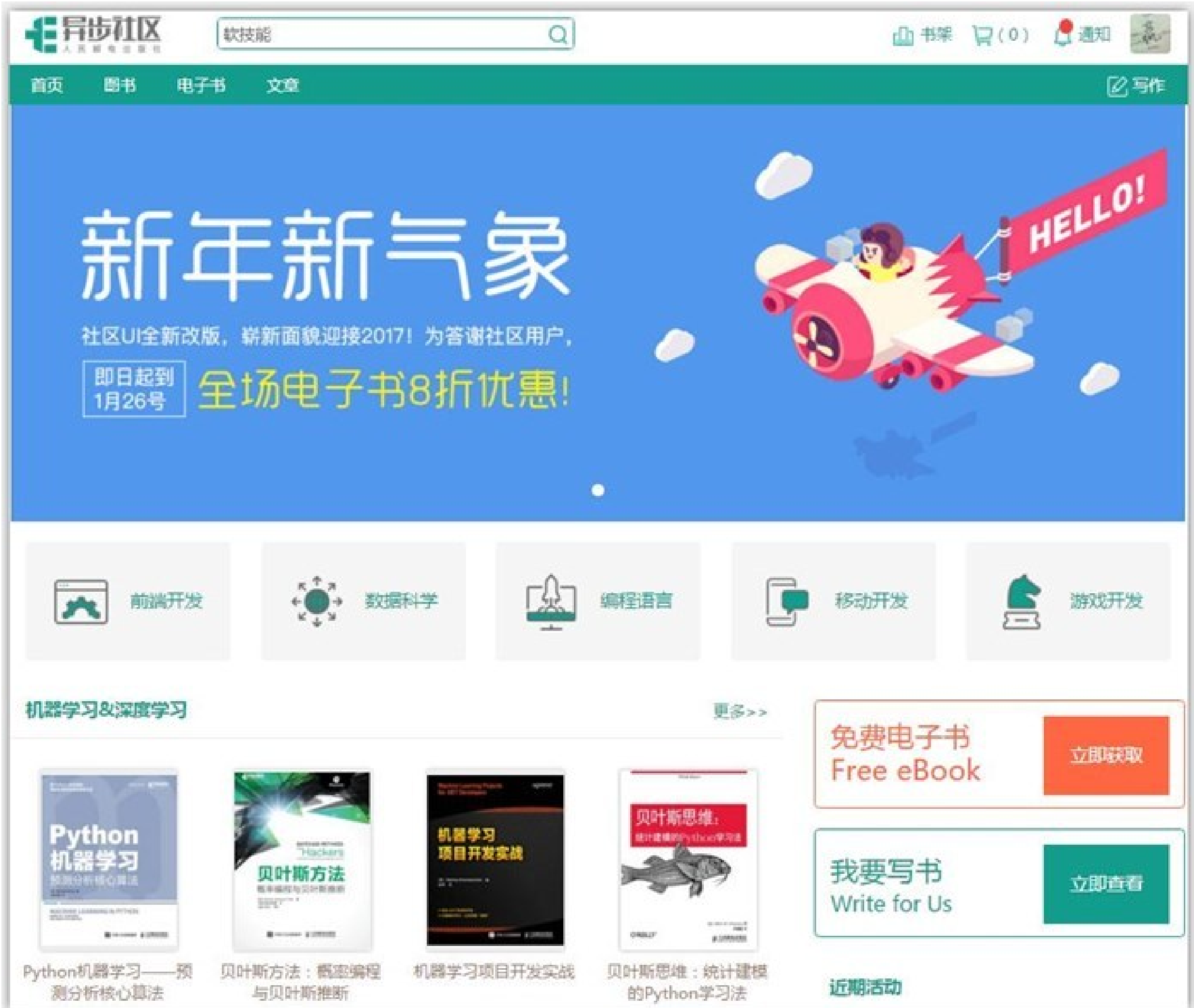
□□□□□□□□□□□□□□□Python Selenium
WebDriver□□□□□□□□□□□□□□□□□Selenium
WebDriver□□Web□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

[illegible]

[illegible]

[illegible](www.epubit.com.cn)[illegible]IT[illegible]2015[illegible]8[illegible]

[illegible]20[illegible]IT[illegible]POD[illegible]



[illegible]

[illegible]

[illegible]IT[illegible]Web[illegible][illegible]1000[illegible]400[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]100[illegible]=1[illegible]

[illegible] [illegible]

[illegible]

[illegible]“**57AWG**”[illegible]“[illegible]”[illegible]8[illegible]

[illegible]

[illegible]

Wireshark网络分析的艺术
作者：林沛满
责编：傅道坤
分类：计算机科学 > 安全与加密 > 网络安全
Wireshark是当前最流行的网络包分析工具。它上手简单，无需培训就可入门。很多
本书挑选的网络包来自真实场景，经典且接地气。讲解时采用了生活化的
更多>>
下载PDF样章
配套文件下载
5.6K 浏览
57 想读
7 推荐
分享：
纸质 ¥45.00 ¥31.50 (7 折)
电子 ¥25.00
电子 + 纸质 ¥45.00
购买
总价：75.60
一起购买
（纸质）
（纸质）
目录
评论 9
勘误 1
出版信息
作者简介
专业书评
内容提要
本书作译者
LinPeiman
上海
1.0K经验值
发私信
送积分
关注
《Wireshark网络分析就这么简单》即《Wireshark网络分析的艺术》作者
兑换样书
立即兑换
如何赚取积分
电子书版本
PDF
Epub
Mobi
精彩推荐
Nmap渗透测试指南
作者：
Nmap

□□□□□□□□□□

□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□100□

□□

□□

[illegible]Markdown[illegible]

[illegible]

[illegible]

[illegible]IT[illegible]

[illegible]

[illegible]

□□□□

□□□□□

□□□□□

□□□□

QQ□□436746675

□□□□□www.epubit.com.cn

□□□□□□□□□□

□□□□□@□□□□□□□□@□□□□□□□□-□□□□□□□

□□&□□□contact@epubit.com.cn